度量指标

发布版本 2021-10

https://chaoss.community/metrics

内容

Common Metrics WG

关注领域	目标
贡献	了解做出了哪些贡献（例如，代码提交，wiki 编辑，文档）。
时间	了解来自组织和个人的贡献何时发生。
人	了解组织和个人参与开源项目。
地点	确定贡献发生在哪些物理和虚拟的地方 (例如，GitHub，聊天平台，论坛，会议)

关注领域 - 贡献

目标: 了解做出了哪些贡献（例如，代码提交，wiki 编辑，文档）。

度量指标	问题
代码克隆（clone）	在本地机器上保存了多少份开源项目代码仓的副本?
编程语言分布	开源项目中有哪些不同的编程语言，每种编程语言所占的百分比是多少?
技术分支 (fork)	在代码托管平台一个开源项目有多少技术分支 (fork)?
贡献类型	正在进行哪些类型的贡献?

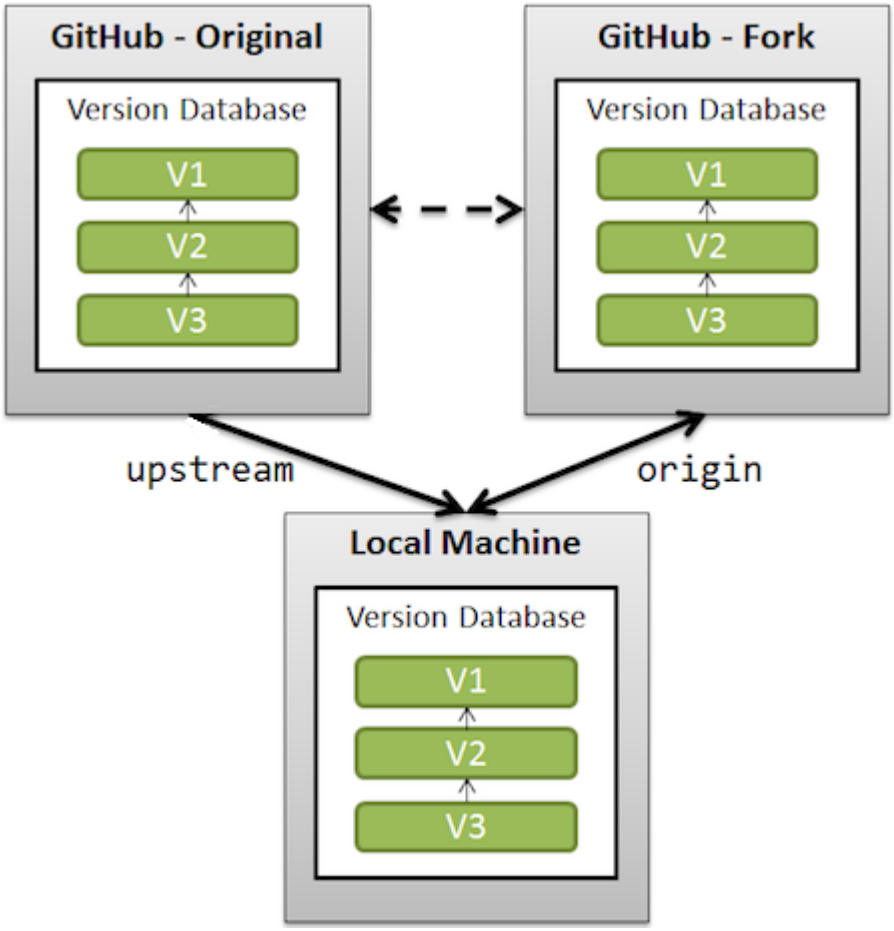
代码克隆（clone）

问题: 在本地机器上保存了多少份开源项目代码仓的副本?

描述

代码克隆（clone）是保存到本地计算机的代码仓的副本。

注意: 许多时候代码克隆（clone）和 技术分支（technical fork） 可以互换使用，但是两者之间是有区别的。一个 技术分支（technical fork） 是同一代码托管平台上代码仓的副本, 而代码克隆（clone）是本地计算机上的副本。



图片来源于 Stakeoverflow

目标

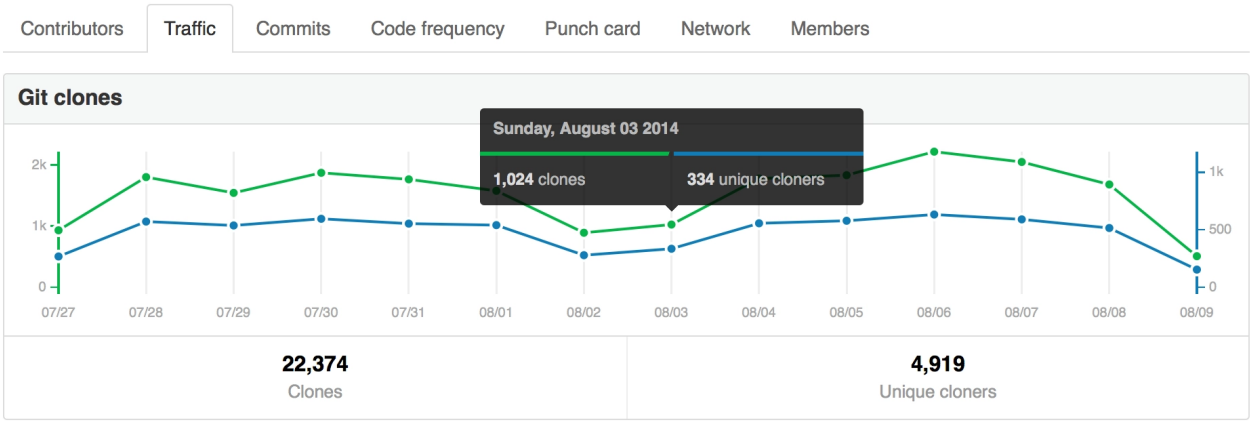
代码克隆（clone）指标的目标是确定一个项目的下载数量。对代码克隆（clone）的分析可以帮助我们深入了解这个项目的受欢迎程度和使用情况。

实现

过滤条件

- 时间周期 (例如, 按星期、月份、年份分的代码克隆（clone）量)
- 部分代码克隆（clone）(由 gitlab 提供 https://docs.gitlab.com/ee/topics/git/partial_clone.html#partial-clone)

可视化效果



图片来源于 Github 博客

参考资料

- 分叉和克隆代码仓之间的区别是什么?
- 分叉和克隆的区别是什么?
- Github API

贡献者

- Vinod Ahuja
- Sean Goggins
- Matt Germonprez
- Kevin Lumbard
- Dawn Foster
- Elizabeth Barron

编程语言分布

问题: 开源项目中有哪些不同的编程语言，每种编程语言所占的百分比是多少?

描述

项目中所使用编程语言的数量和每种语言所占的百分比可以使人们对代码贡献者所需要的技能以及项目本身的性质有所了解。

目标

这个度量标准将帮助特定开源项目的新手，同时也为开源项目经理结合他们自己的经验和组织，提供了一个关于项目概况的视角。

在关注价值度量的领域中: 作为求职的一部分，开发人员可以使用这个指标来识别那些深度依赖于他们所使用的编程语言的项目。

在关注演进度量的领域中: 这个指标可以用于标识随着时间的推移每种语言中文件或代码行数量的变化。例如，一个项目在某个时间点可能是 x% Python 和 y% Javascript。也许一年后，由于项目的重点转向了用户体验，用文件数或代码行数来衡量的 Javascript 使用量可能会更多。

在关注风险度量的领域中: 这个度量可以与上游依赖度量相结合，以确定是否在项目中使用了某个重要的语言，但尚未被依赖扫描工具确认。

在关注多样，平等与包容度量的领域中: 当包容性的、多样化的和公平的社区被确定后，它们将有一定程度的编程语言分布。

当一个人寻找新的项目时，知道哪些项目依赖于他们已经知道的或者想要学习的编程语言，可以是一系列 “个人过滤器” 中的一个。

一般来说，这个度量对开源办项目公室是有用的，社区管理人员的目标是了解哪种语言最突出，哪种语言很少被使用，但是很重要。

实现

编程语言分布使用一种先验的可识别计算机编程语言来考虑代码仓中每个文件的不同属性。当新语言的出现，可能会出现计数工具无法识别其文件后缀的初始阶段，在这种情况下，它们可能被视为 “其他” 语言。这样的时期通常是短暂的。

过滤条件

- 时间
- 文件数 - 每种语言的文件数。

编程语言	文件数
Python	246
JSON	28
BASH	26
Plain Text	14
Shell	12
gitignore	8
YAML	6
Makefile	4
R	2

编程语言	文件数
Docker ignore	2
Dockerfile	2
Markdown	2

- 代码行数 - 每种语言代码行数的百分比.

编程语言	每种语言代码行数的百分比
Python	94.60%
BASH	2.48%
Shell	1.08%
JSON	0.52%
R	0.42%
Plain Text	0.38%
Makefile	0.16%
YAML	0.15%
gitignore	0.12%
Dockerfile	0.05%
Docker ignore	0.03%
Markdown	0.00%

代码行数或文件数都可以表示为绝对数或百分比，具体取决于度量所使用的应用程序。在许多情况下，简单的文件计数是有用的，而代码的绝对行数可能很难区分，因为量级要大得多。

提供度量的工具

- Augur-Community-Reports 代码仓目前提供这个指标。
- GrimoireLab 通过识别文件后缀名来提供这个指标。
- Augur 在前端提供，同时也通过 API endpoint 提供.

数据收集策略

代码仓的内容可以通过迭代每个文件来计算，这种方案有几个库支持实现: https://github.com/boyter/scc

一些编程语言的文件扩展名，比如 Jupyter Notebooks，可能会被排除在外，因为它们掩盖了实际使用的编程语言。

参考资料

- https://github.com/boyter/scc

贡献者

- Dawn Foster
- Beth Hancock
- Matt Germonprez

- Elizabeth Barron
- Daniel Izquierdo
- Kevin Lumbard
- Sean Goggins

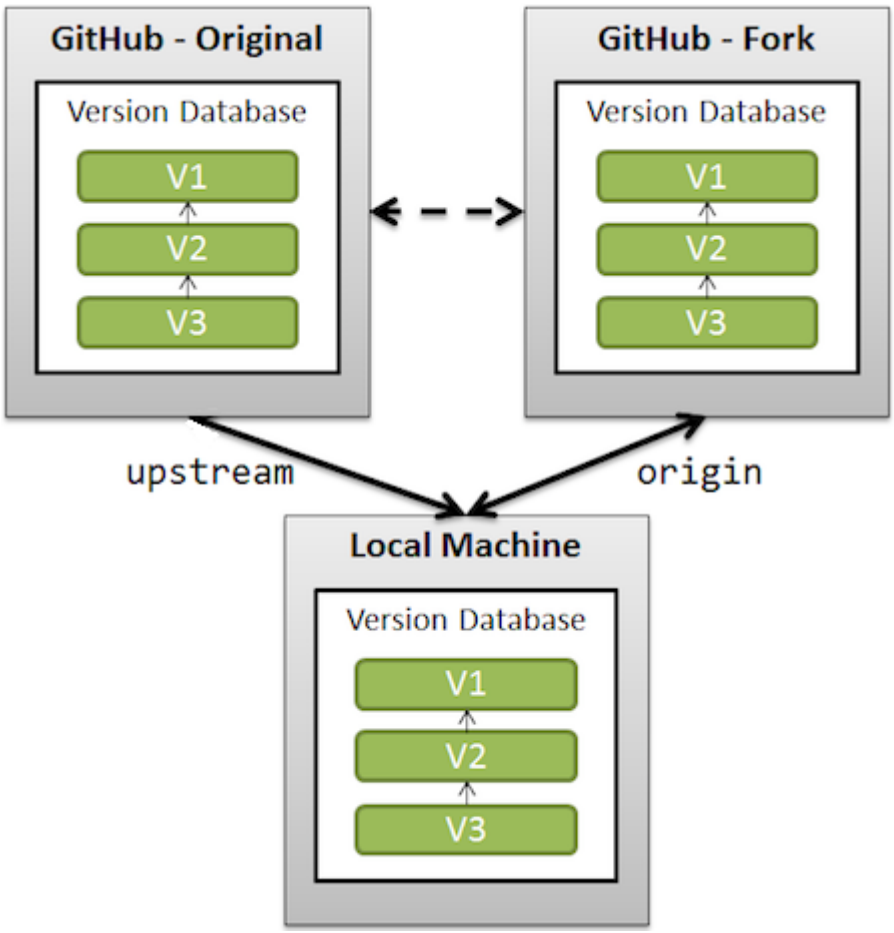
技术分支 (fork)

问题: 在代码托管平台一个开源项目有多少技术分支 (fork)?

描述

一个技术分支 (fork) 是一个项目分布式版本控制的副本。技术分支 (fork) 数表示在同一代码开发平台上该项目的副本数。

注意: 许多时候代码克隆（clone）和 技术分支（technical fork） 可以互换使用，但是两者之间是有区别的。一个 技术分支（technical fork） 是同一代码托管平台上代码仓的副本, 而代码克隆（clone）是本地计算机上的副本。



图片来源于 Stakeoverflow

目标

技术分支 (fork) 度量的目标是确定在代码开发平台上存在的该项目的副本数。对技术分支 (fork) 的分析可以提供对分支意图的深入了解 (建立分支的不同意图，如对上游有贡献的分支和没有贡献的分支)。

实现

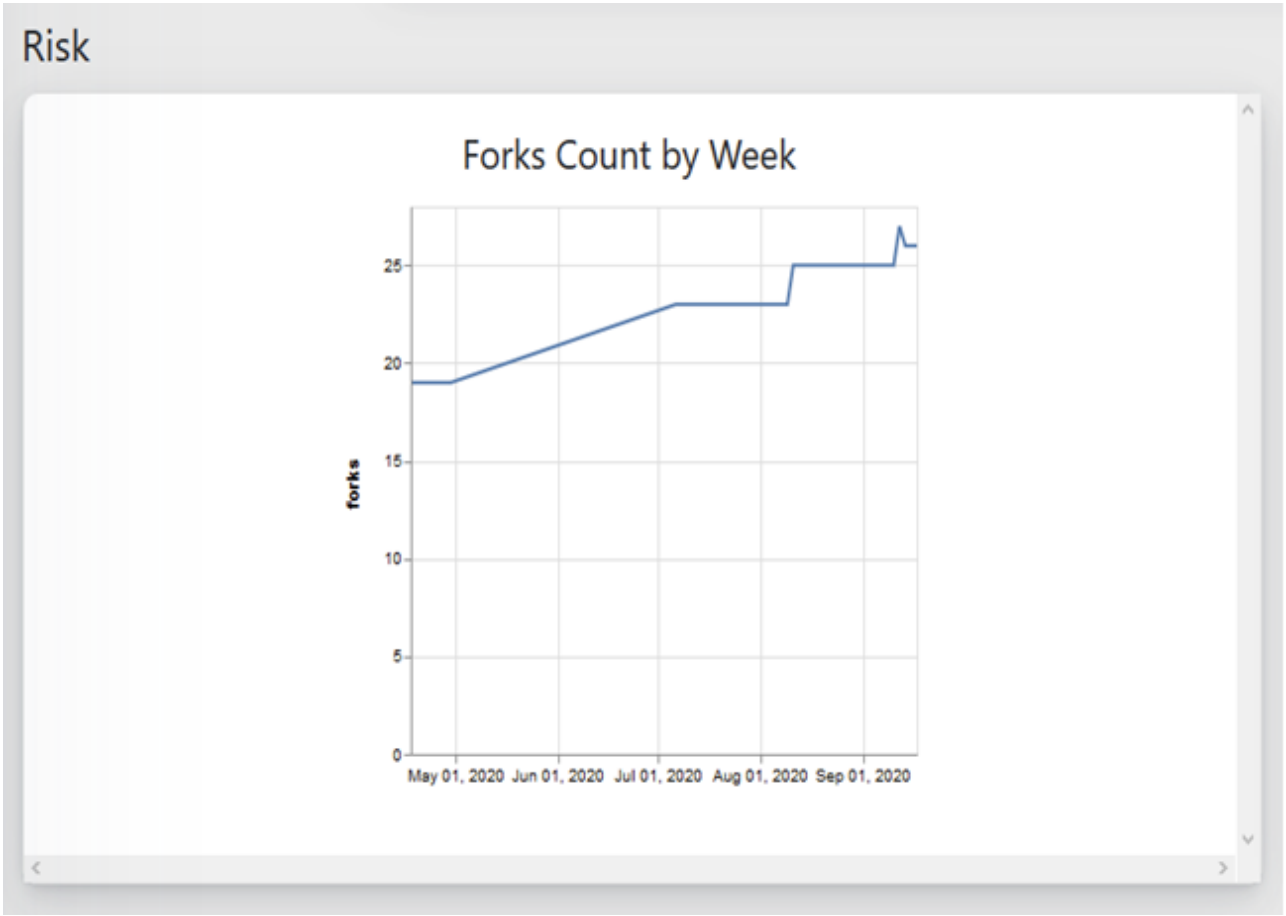
筛选条件

- 时间周期 (例如，每周，每月，每年)
- 贡献分支与总分支的比率 (贡献分支是指对上游代码仓库建立更改请求的分支。)

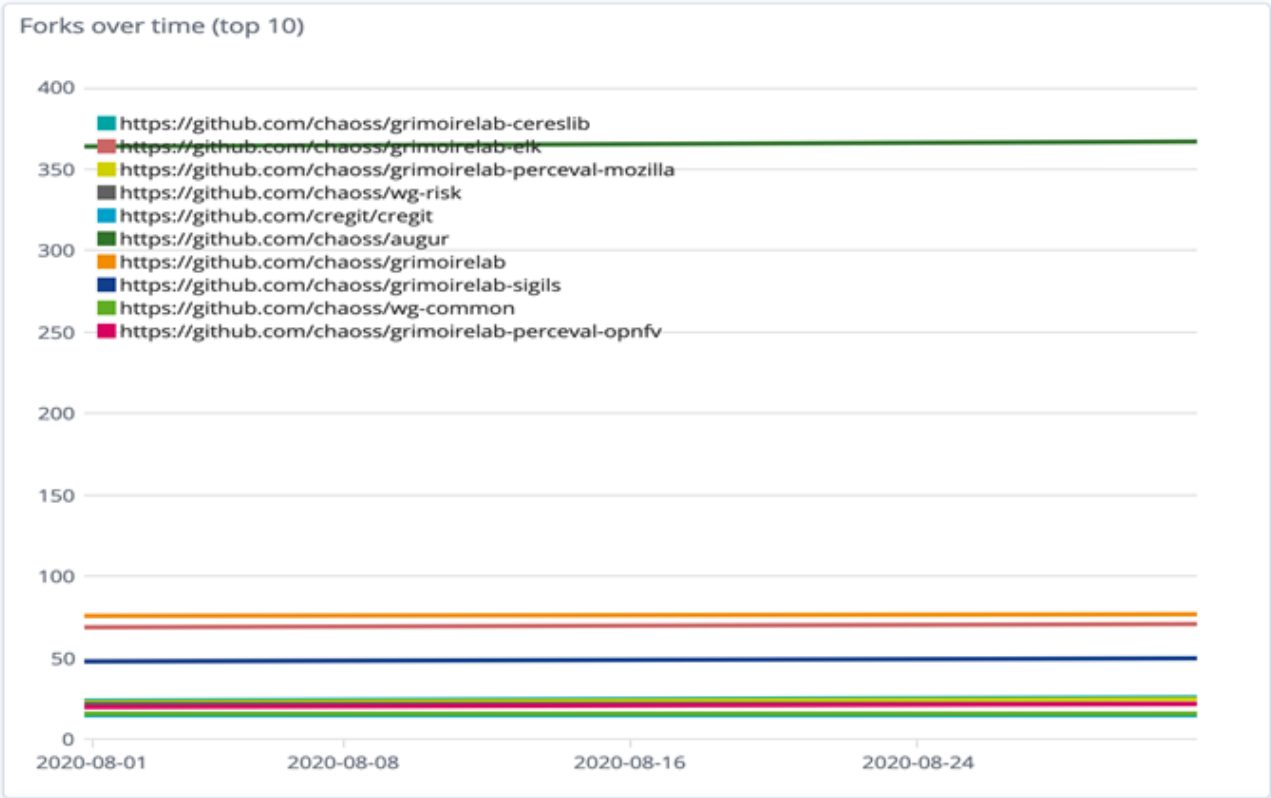
- 非贡献分支与总分支的比率 (贡献分支是指从未对上游代码仓库建立更改请求的分支。)

可视化效果

Augur 实现



GrimoireLab 实现



提供度量的工具

- Augur
- GrimoireLab

数据收集策略

Github API
https://developer.github.com/v3/repos/forks/#list-forks

GitLab API
https://docs.gitlab.com/ee/api/projects.html#list-forks-of-a-project

Bitbucket API
https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Brepo_slug%7D/forks

参考资料

https://help.github.com/en/enterprise/2.13/user/articles/fork-a-repo https://opensource.com/article/17/12/fork-clone-difference

贡献者

- Vinod Ahuja
- Sean Goggins
- Matt Germonprez
- Kevin Lumbard

- Dawn Foster
- Elizabeth Barron

贡献类型

问题：正在进行哪些类型的贡献？

描述

多元化的贡献能够使开源项目健康发展。在很多项目中，有些社区成员并不编写代码，但他们同样做出了有价值的贡献，比如管理社区、区分错误、宣传项目、支持用户或以其他方式提供帮助。

目标

多样的贡献类型表明项目是成熟全面的，包含足够的活动来支持项目的所有方面，并且提供多种贡献类型的晋升渠道，让拥有编码之外的不同专长的人员也能够发展到领导层。

实现

如何对贡献进行定义、量化、跟踪和公布是一个具有挑战性的问题。每个项目的答案可能都是独一无二的，以下建议仅作为抛砖引玉。作为一个通用指南，很难将不同的贡献类型相互比较，应单独考量。

- 以下列表可以帮助确定贡献类型：
    - 编写代码
    - 审查代码
    - 错误分类
    - 质量保证和测试
    - 安全相关活动
    - 本地化和翻译
    - 事件组织
    - 文档编写
    - 社区建设和管理
    - 教学和教程构建
    - 故障排除和支持
    - 创意作品和设计
    - 用户界面、用户体验和易用性
    - 社交媒体管理
    - 用户支持和问题解答
    - 撰写文章
    - 公共关系 - 技术媒体采访
    - 事件发言
    - 营销与活动宣传
    - 网站开发

    - 法律委员会
    - 财务管理

数据收集策略

- ** 采访或调查：** 让社区成员认可他人的贡献，找出过去没有考虑到的贡献类型。
    - 您想表彰谁在项目中的贡献？他们贡献了什么？
- ** 观察项目：** 找出和认可项目不同部分的负责人。
    - 在项目网站或代码仓库中列出了哪些负责人？
- ** 捕获非代码贡献：** 通过问题跟踪器等专用系统跟踪贡献。
    - 日志记录可以包含项目要了解的关于非代码贡献的定制化信息，用以识别工作量。
    - 通过沟通渠道活动的代理贡献。例如，如果质量保证成员 (QA) 拥有自己的邮件列表，就可以通过邮件列表活动来代理衡量围绕 QA 贡献的活动。
- ** 收集跟踪数据：** 通过协作工具日志数据衡量贡献。
    - 例如，可以从源代码仓库计算代码贡献，可以从维基编辑历史记录计算维基贡献，可以从电子邮件归档计算电子邮件消息
- ** 自动分类：** 训练人工智能 (AI) 机器人来检测贡献并对其分类。
    - AI 机器人可以协助对贡献进行分类，例如，帮助请求与提供的支持，或功能请求与错误报告，尤其是以上均在同一个问题跟踪器中完成的情况。

其他考量：

- 特别是对于自动报告，允许社区成员选择退出并不出现在贡献报告上。
- 承认对贡献类型的捕捉不完善，并明确说明收集了哪些类型的贡献。
- 随着项目发展，贡献类型的收集方法需要作出调整。例如，交换国际化库时，围绕本地化的项目活动可能会在变化前后产生不同的指标。
- 大规模挖掘贡献类型时，要考虑机器人的活动。

参考资料

- https://medium.com/@sunnydeveloper/revisiting-the-word-recognition-in-foss-and-the-dream-of-open-credentials-d15385d49447
- https://24pullrequests.com/contributing
- https://smartbear.com/blog/test-and-monitor/14-ways-to-contribute-to-open-source-without-being/
- https://wiki.openstack.org/wiki/AUCRecognition
- https://www.drupal.org/drupalorg/blog/a-guide-to-issue-credits-and-the-drupal.org-marketplace

关注领域 - 时间

目标: 了解来自组织和个人的贡献何时发生。

度量指标	问题
活跃的日期和时间	贡献者活跃的日期和时间戳是什么时候?
突发性波动	如何在项目中观察到一个活动在短时间的突发性波动，然后相应地恢复到这个活动的正常状态?
变更请求中审查周期的持续时间	单个变更请求中审查周期的持续时间是多少?
第一响应时长	对于需要关注的活动，从创建到第一次响应需要多长时间?
关闭时长	创建和关闭操作（如议题、更改请求或需要支持的问题单）之间需要多少时间?

活跃的日期和时间

问题：贡献者活跃的日期和时间戳是什么时候？

描述

个人在一天的不同时间参与开源项目活动。此指标旨在确定个人活动完成的日期和时间。在非 UTC 时区，数据可用于概率性估计贡献来自地球哪个地方。

目标

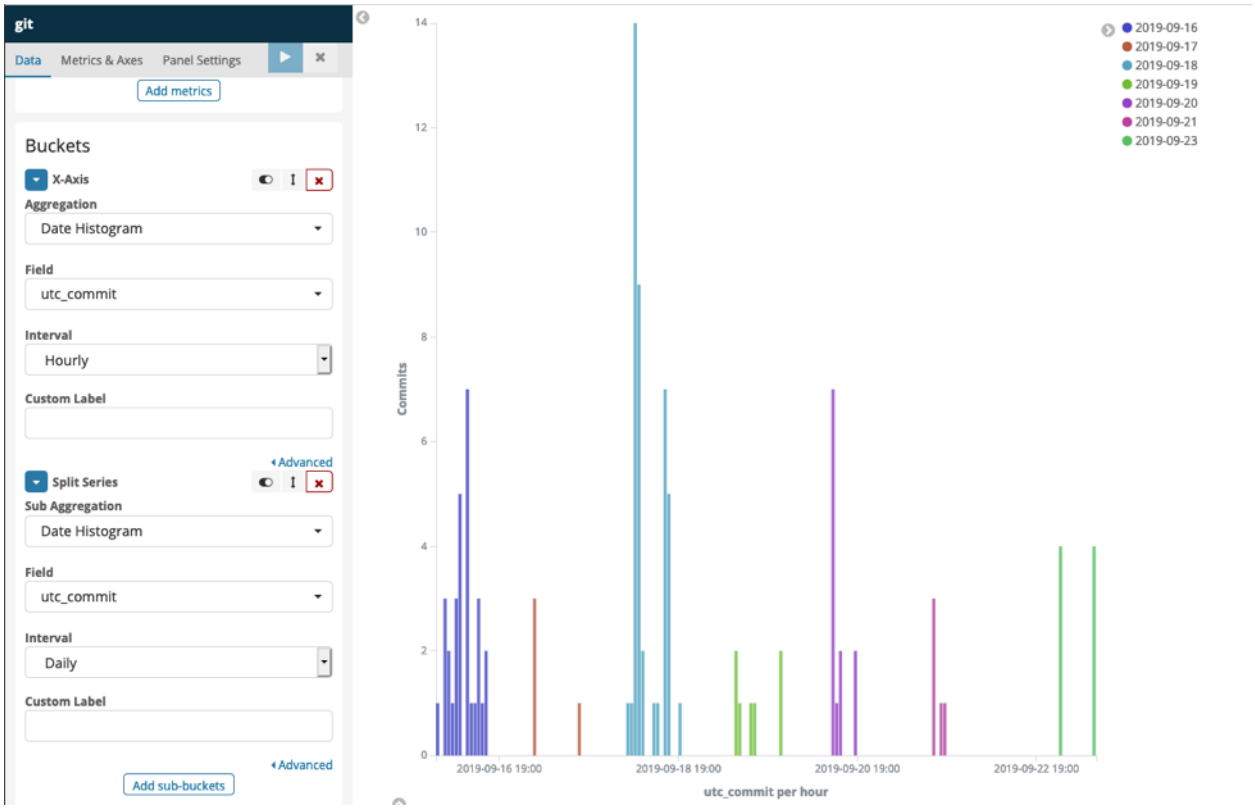
- 为雇主提高透明度，帮助其了解组织员工参与开源项目的时间
- 为开源项目和社区管理者提高透明度，帮助其了解活动发生的时间

实现

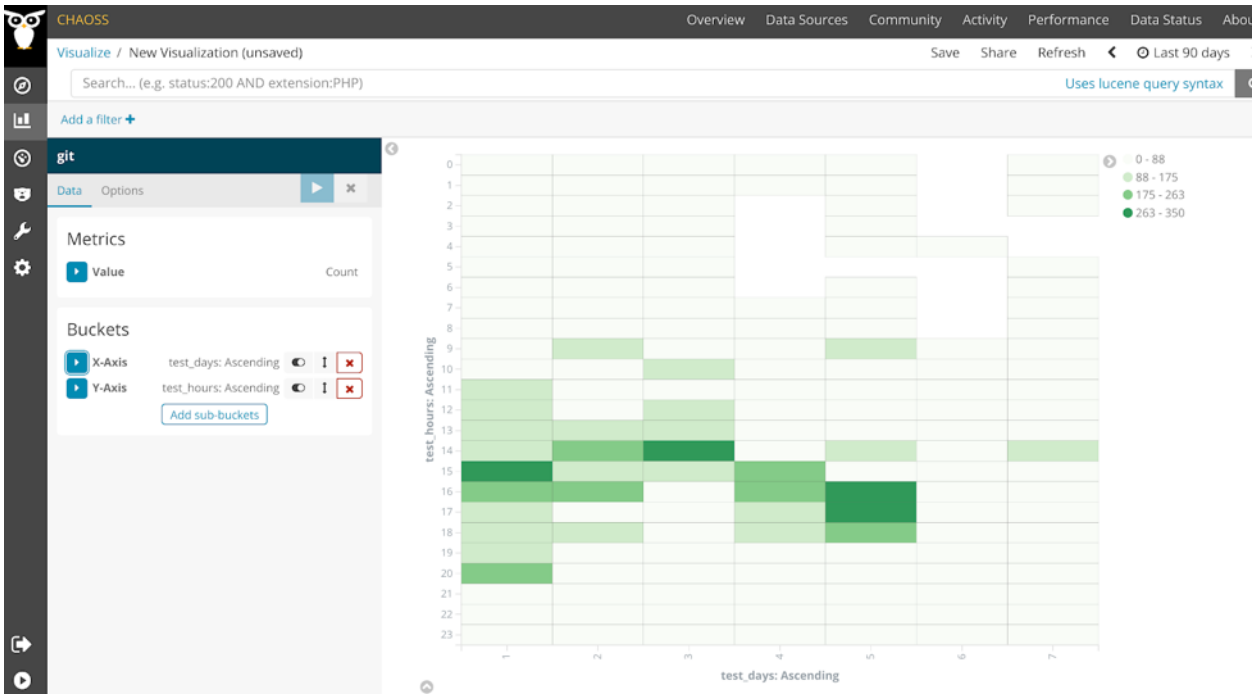
筛选条件

- 通过组织筛选个人
- 按 UTC 时间聚合
  - 可以显示全球贡献时间分布；项目最活跃时间。
- 按当地时间聚合
  - 可以显示贡献者做贡献时的当地时间。可以得到结论贡献者是在工作时间贡献更多，还是在晚间（非工作时间）贡献更多。
- 代码仓库 ID
- 社区细分（例如，GrimoireLab 在欧盟时区更活跃，Augur 在美国时区更活跃）

可视化效果



Time	author_name	repo_name	tz	utc_author	utc_commit	commit_date
Sep 23rd 2019	Carter Landis	https://github.com/chaoss/augur	-5	September 23rd 2019, 15:42:16.000	September 23rd 2019, 15:42:16.000	September 23rd 2019, 10:42:16.000
Sep 23rd 2019	Matt Snell	https://github.com/chaoss/augur	-5	September 23rd 2019, 13:21:56.000	September 23rd 2019, 13:21:56.000	September 23rd 2019, 08:21:56.000
Sep 23rd 2019	Gabe Heim	https://github.com/chaoss/augur	-5	September 23rd 2019, 12:31:49.000	September 23rd 2019, 12:31:49.000	September 23rd 2019, 07:31:49.000
Sep 23rd 2019	Santiago Dueñas	https://github.com/chaoss/grimoirelab-perceval	2	September 23rd 2019, 12:30:18.000	September 23rd 2019, 12:30:18.000	September 23rd 2019, 14:30:18.000
Sep 23rd 2019	Valerio Cosentino	https://github.com/chaoss/grimoirelab-perceval	2	September 23rd 2019, 11:26:04.000	September 23rd 2019, 11:26:30.000	September 23rd 2019, 13:26:30.000
Sep 23rd 2019	Valerio Cosentino	https://github.com/chaoss/grimoirelab-perceval	2	September 23rd 2019, 11:24:21.000	September 23rd 2019, 11:24:21.000	September 23rd 2019, 13:24:21.000



提供指标的工具

GrimoireLab

Augur 日期/时间戳

参考资料

协调世界时

突发性波动

问题: 如何在项目中观察到一个活动在短时间的突发性波动，然后相应地恢复到这个活动的正常状态?

描述

有多种原因可能会促使代码仓库中的某个活动的总量突然增加或减少。这些增加和减少都表现为该活动相对于平均活动量的突然变化。突发性是一种了解现有指标中活动周期的方法，例如问题、合并请求、邮件列表、代码提交或评论。某个活动突发性变化的根本原因的示例包括:

- 发布周期
- 全球流行病
- 黑客松活动
- 导师计划
- 展示工具的会议、聚会和其他活动
- 传统和社交媒体的公告和提及
- 引起人们高度关注的关键 Bug
- 为解决特定问题的社区设计会议或头脑风暴会议
- 社区成员来自于另一个依赖你项目 (例如，依赖项) 的社区

目标

- 确定活动突发性波动的根本原因的影响
- 当项目活动在不知不觉中上升时提高意识
- 帮助捕捉某个项目活动增加或减少的意义
- 帮助社区和维护者为遵循某种规律的未来突发性波动做好准备
- 帮助衡量有影响力的外部活动对本社区的影响
- 区分非正常活动与正常活动

实现

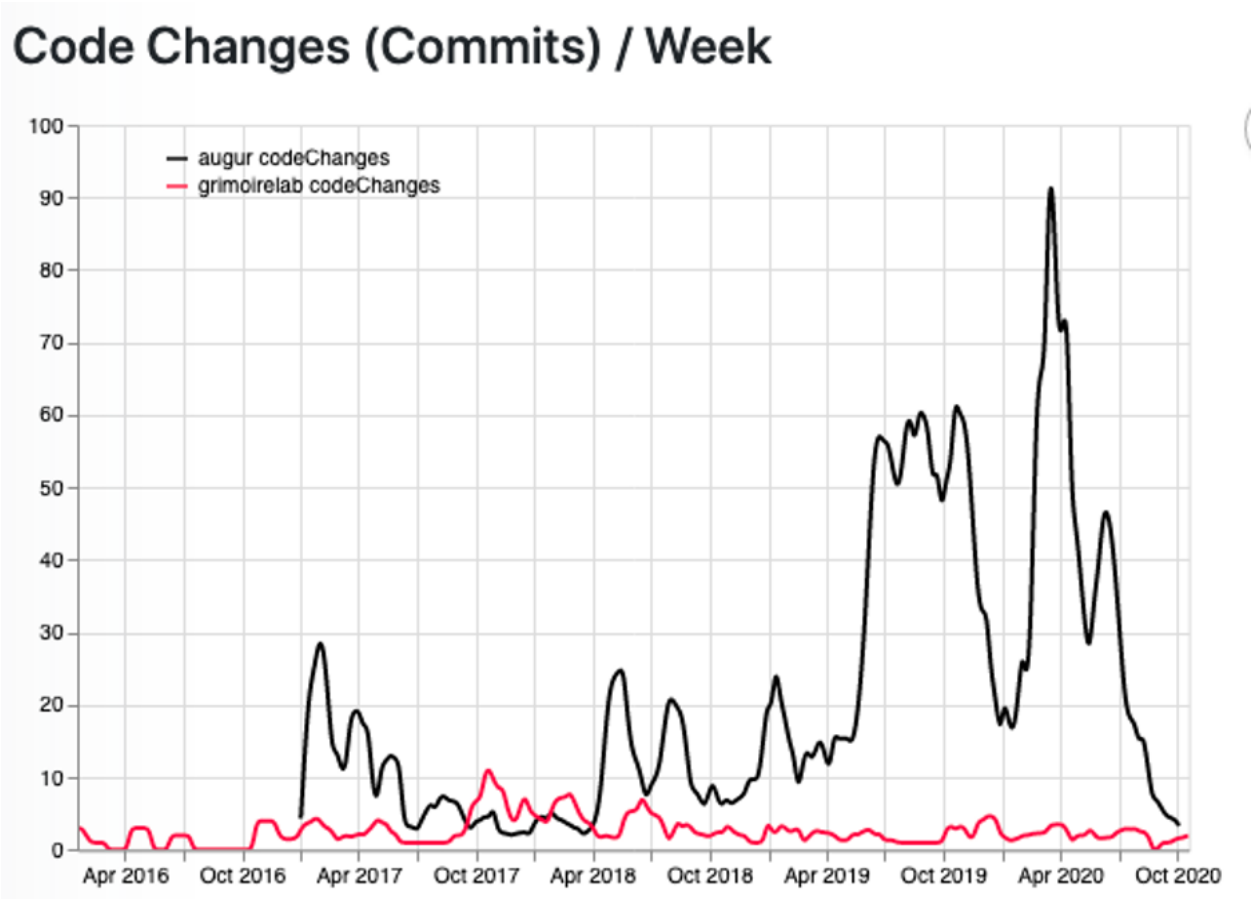
筛选条件

- 标星
- 分支
- 问题或者 bug 报告
- 标签
- 下载
- 发布标签
- 变更请求

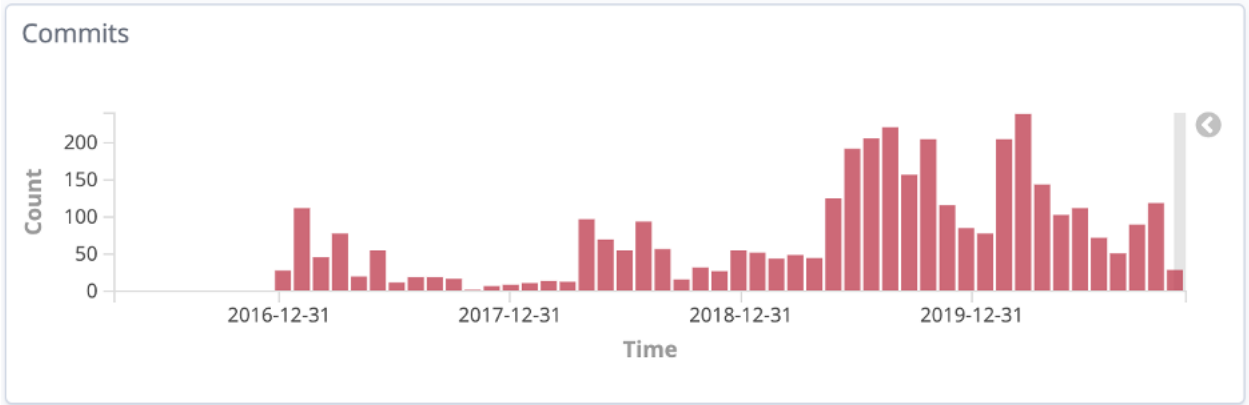
- 邮件列表流量
- 文档添加或修订
- 新建代码仓库
- 新功能请求
- 消息对话
- 传统和社交媒体活动
- 会议出席和提交

可视化效果

Augur:



GrimoireLab:



提供度量的工具

- Grimoire Lab
- Augur

数据收集策略

- 定量
  - 基于时间框识别偏离某种常规状态的活动
  - 某些阈值的异常值，使用布林带等统计数据来衡量稳定性或波动性: https://en.wikipedia.org/wiki/Bollinger_Bands
- 定性调查问题
  - 为什么你在某一段时间内贡献量更多?
  - 你认为这些特有突发性波动的根本原因是什么?
  - 不同事件 (例如, 黑客松，导师计划，或者会议) 对项目活动有什么影响?

References

该指标的灵感来自 Goh 和 Barabasi (2008): https://arxiv.org/pdf/physics/0610233.pdf

变更请求中审查周期的持续时间

问题: 单个变更请求中审查周期的持续时间是多少?

描述

一个变更请求会基于一个或者多个审查周期。在一个审查周期内，一个或多个审查者可以对提议的贡献提供反馈。审查周期的持续时间，或贡 献的每次新迭代之间的时间，是该指标的基础。

目标

该项目为项目维护者提供的相关洞察:

- 代码审查过程迟滞缓慢, 因为有更多的的迭代以及审查周期增加。
- 流程瓶颈导致代码审查迭代时间长。
- 审查周期中维护者或提交者响应缓慢，导致审查周期处于放弃或者半放弃状态。
- 具有不同审查循环周期长度的评论的特征。

实现

审查周期持续时间用于衡量单个变更请求中一个审核周期的时间长度。持续时间可以计算的间隔包括:

- 每个审查周期开始的时刻，定义为提交或更新变更请求的时间点。
- 每个审查周期结束的时刻，可能是因为变更请求已更新并需要重新审查，或者是因为变更请求被接受或拒绝。

筛选条件

平均或中位值持续时间, 可以选择性的使用过滤或者分组条件包括:

- 参与审核的人数
- 审核中的评论数
- 变更请求中涉及的修改文件或者代码
- 提出变更请求的项目或组织
- 提交变更请求的时间
- 为变更请求做出贡献的开发人员
- 变更请求
- 变更请求的审核周期数（例如，按第一、第二、.....轮过滤）

可视化效果

提供度量的工具

参考资料

可用于实现度量的数据示例: https://gerrit.wikimedia.org/r/c/mediawiki/core/+/194071

第一响应时长

问题：对于需要关注的活动，从创建到第一次响应需要多长时间?

描述

对活动的首次响应有时可能是最重要的响应。首次响应表明社区活跃并且积极参与沟通。如果对活动的响应时间较长，可能表示社区不够积极。短时间内对活动做出响应，有助于吸引更多成员参与社区内的进一步讨论。

目标

确定各种活动的首次响应节奏，涉及 PR、问题、电子邮件、IRC 帖子等。对于项目的新贡献者、长期贡献者以及整体项目健康情况，首次响应时长都是一项重要考虑因素。

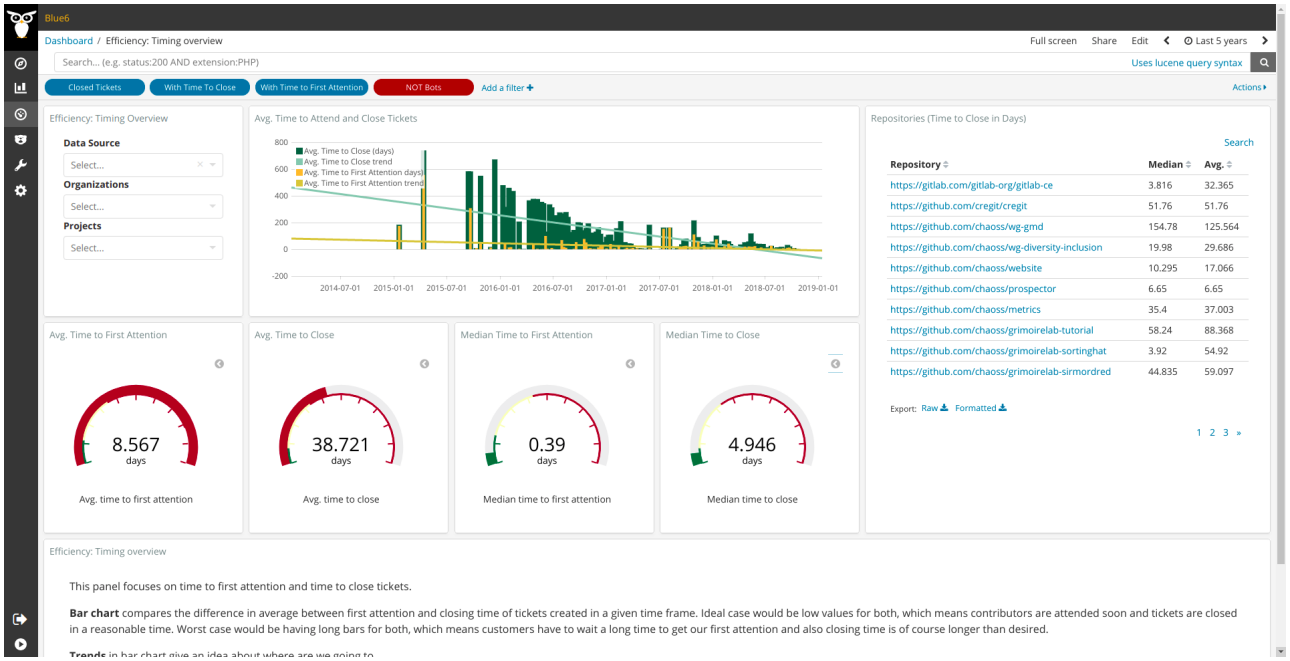
实现

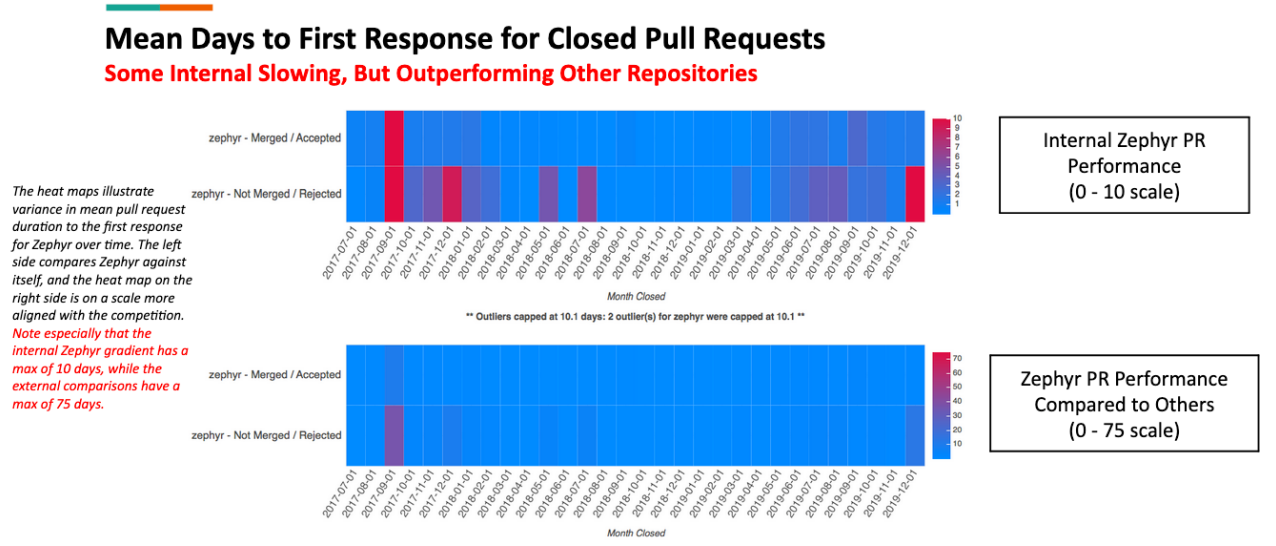
活动的首次响应时长 = 首次响应发布到活动的时间 - 活动创建时间。

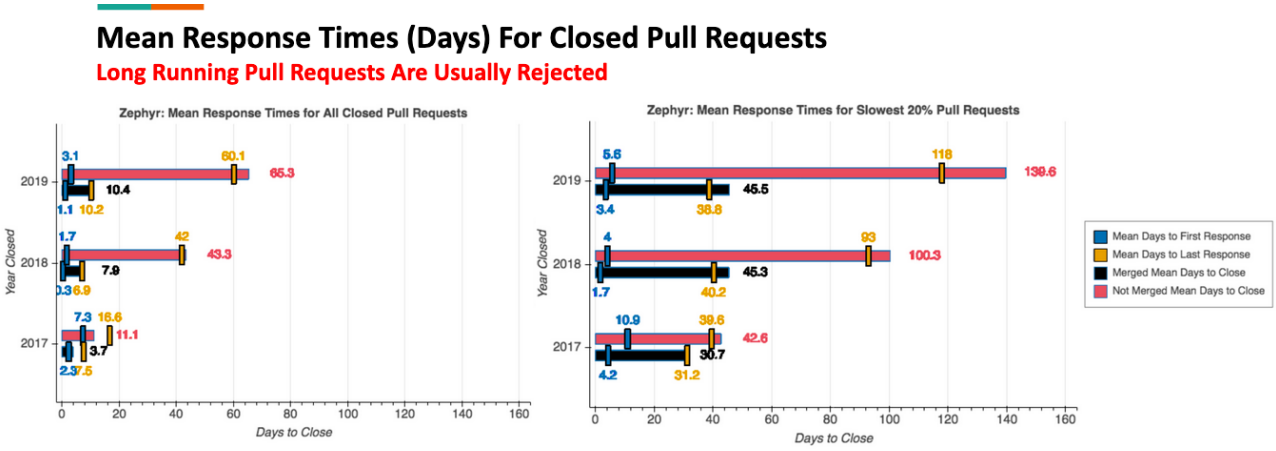
筛选条件

- 回复者的角色，例如，只计算维护者的响应
- 自动响应，例如，通过筛选排除机器人和其他自动回复，只统计真人回复
- 活动类型，如问题（见指标问题响应时间）、电子邮件、聊天信息、变更请求

可视化效果







提供指标的工具

- GrimoireLab 面板：效率时序总览
- Kata Containers 仪表板效率面板

参考资料

关闭时长

问题：创建和关闭操作（如议题、更改请求或需要支持的问题单）之间需要多少时间？

描述

关闭时长是指从创建到关闭操作（如议题、更改请求或需要支持的问题单）的总时长。操作需要具有打开和关闭的状态，比如代码审查进程、问答论坛、问题跟踪系统中经常出现的情况。

相关指标：问题解决时长

目标

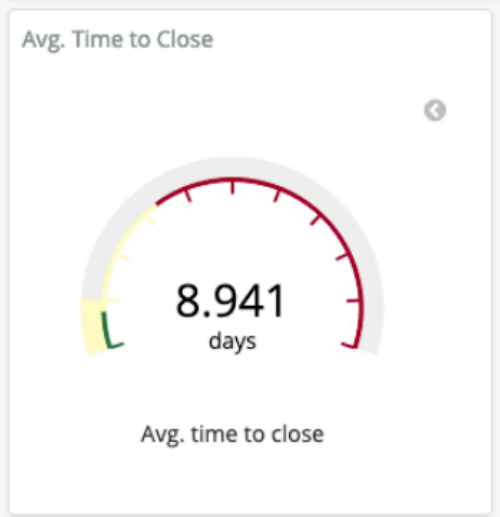
1. 确定社区的响应程度，帮助增加包容性，吸引新贡献者并保留现有贡献者。
2. 找出导致快速或缓慢关闭的操作特征（如寻找最佳实践、改进领域、评估效率）。
3. 识别对不同社区成员及时响应的偏差。
4. 检测社区活动的变化（例如，显示潜在的维护者倦怠、贡献多元化的减少）
5. 了解关闭议题或更改请求的时间与合并成功或失败的关系

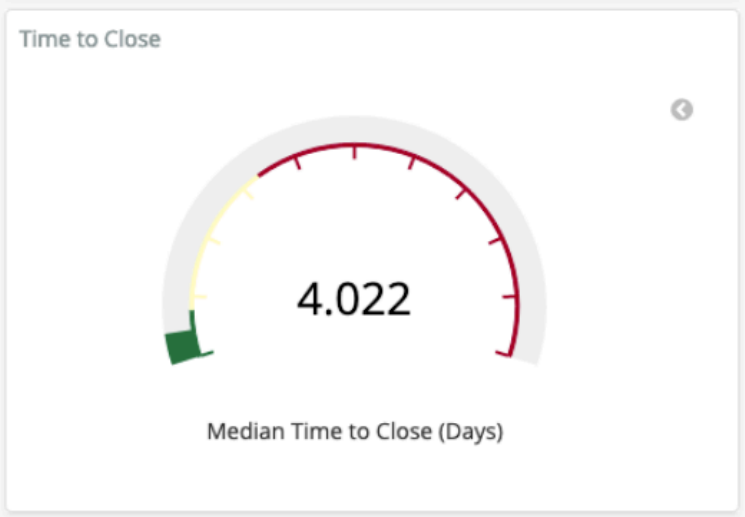
实现

筛选条件

- 操作的创建者（例如，新贡献者相对于维护者）
- 最初关闭，最后关闭
- 标签（例如错误与新功能）
- 更改请求合并状态（例如，合并时间或没有合并的关闭时间）

可视化效果





提供指标的工具

Augur 实现：

- 问题解决时长
- 问题持续时间
- 问题响应时间

GrimoireLab 实现：

- 拉取请求效率
- 问题效率
- Efficiency:TimingOverview

数据收集策略

关闭时长指标可根据项目活动和目标的具体情况而定。例如，错误报告的关闭时长可能提供与新功能请求的关闭时长不同的信息。数据收集策略应针对不同的项目目标。可能影响这些进程的其他变量是：

- 问题跟踪系统：如错误报告、蓝图 (OpenStack 专有命名)、用户故事（user story）、功能请求、epic 等可能会影响事件关闭时长的议题类型。优先级或严重性等其他变量可能有助于推进这一事件的关闭速度。
- 变更请求流程：这取决于变更请求的平台架构，如 Gerrit、GitHub 或邮件列表（如 Linux 内核中），并可能根据进程的复杂程度而有所不同。例如，新人和经验丰富的高级开发者将以不同的方式开展进程，所需时间或多或少。
- 问答论坛：这取决于回答的质量和提问者的意见。有效答案会被标记，提问者成功找到自己问题的正确答案后，进程随即关闭。

参考资料

- “Practice P.12: Respond to all submissions”，出自 “Appendix to: Managing Episodic Volunteers in Free/Libre/Open Source Software Communities”，Ann Barcomb、Klaas-Jan Stol、Brian Fitzgerald 和 Dirk Riehle： https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/13519

关注领域 - 人

目标: 了解组织和个人参与开源项目。

度量指标	问题
机器人（bot）活动	自动化机器人活动的数量是多少?
贡献者	谁是项目的贡献者?
贡献者位置	贡献者的位置在哪里?
组织多元化	什么是贡献的组织多元化?

机器人（bot）活动

问题: 自动化机器人活动的数量是多少?

描述

机器人是一种支持项目活动的软件。机器人在开源项目中扮演着越来越重要的角色，帮助协调开源项目的工作。机器人支持各种工作流程，包括议题和合并请求管理和贡献者协议。机器人为开源软件项目提供了许多选择，以决定如何有效地管理项目工作。

目标

机器人活动指标有助于区分人和自动化应用程序（例如，机器人）的项目活动。这样一来，社区管理者可以更好地区分新贡献者的增加、CLA 机器人活动和机器人相关的访问控制等。这个指标可以为以下方面提供额外的背景。

- 首次响应时间
- 构建处理时间
- 社交考虑 -- 不是所有机器人
- 成熟度
- 社区大小
- 项目透明度

实现

- 随着时间推移，机器人与人类活动的比率
- 随着时间的推移，机器人的平均数量

过滤条件

- 具有用户档案的机器人
- 需要人工交互的机器人
- 自行运行的机器人
- 协助软件开发的机器人
- 协助沟通的机器人
- 协助访问控制的机器人
- 协助包容性的机器人
- 使用机器人的平台 (例如, GitHub, Slack)

可视化效果

image

From: https://k8s.devstats.cncf.io/d/5/bot-commands-repository-groups?orgId=1&var-period=w&var-repogroup_name=Kubernetes&var-repo_name=kubernetes%2Fkubernetes&var-commands=All

参考资料

- 6 个可以帮助你开源项目的机器人
- GiHub 的 Hubot

贡献者

- Sean Goggins
- Matt Germonprez
- Kevin Lumbard
- Dawn Foster
- Elizabeth Barron
- Vinod Ahuja
- Matt Cantu

贡献者

问题：谁是项目的贡献者?

描述

贡献者是指以任意方式为项目做出贡献的人。这一指标确保所有类型的贡献在项目中都能得到充分认可。

目标

开源项目由许多贡献者组成。认识项目的所有贡献者对于了解个人参与的活动非常重要，比如谁在帮助代码开发、事件规划和营销工作等。

实现

从项目使用的协作工具中收集作者姓名。

聚合器：

- 计数。给定时间内的贡献者总数。

参数：

- 时间段。开始日期和完成日期。默认：永久。计算贡献的时期段。

筛选条件

按参与地点。例如：

- 提交作者
- 议题作者
- 审查参与者，例如拉取请求中
- 邮件列表作者
- 事件参与者
- IRC 作者
- 博客作者
- 按发布周期
- 项目活动的时间框架，例如，寻找新贡献者
- 项目的编程语言
- 项目中的角色或职能

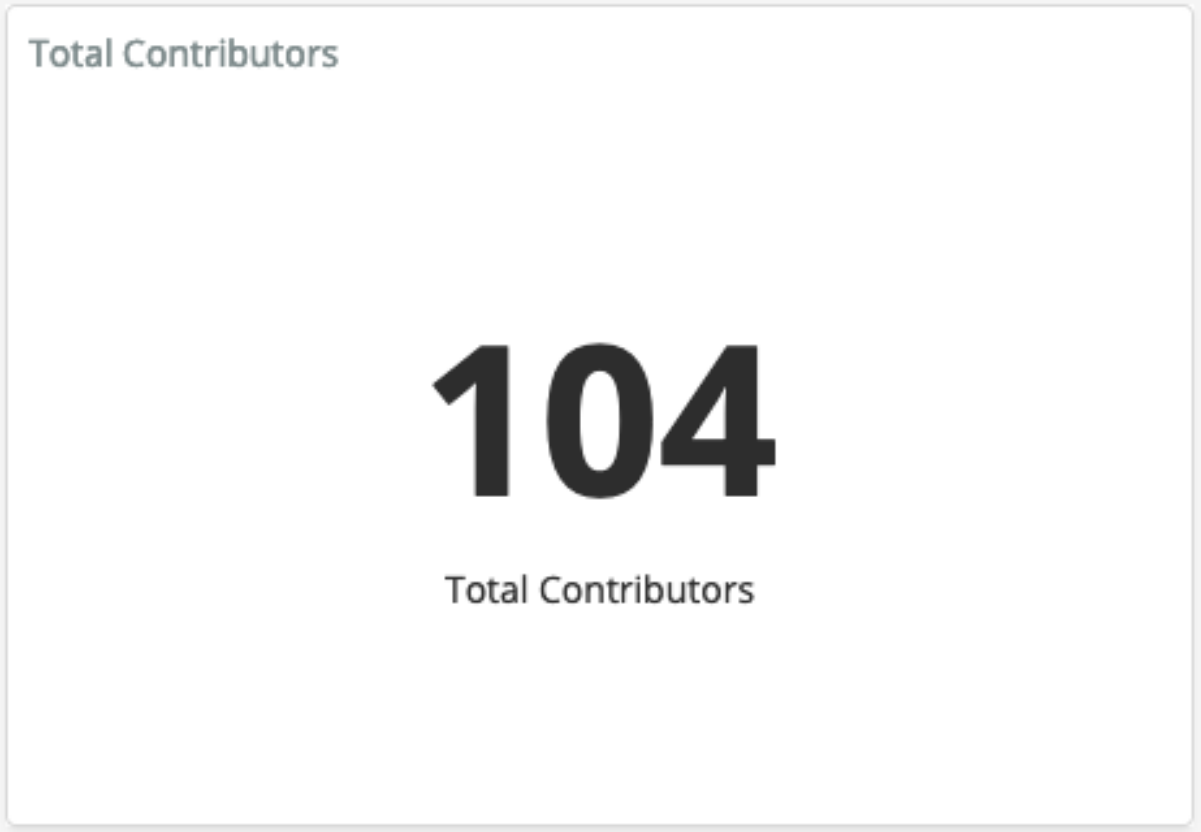
可视化效果

1. 贡献者名称列表（通常带有参与程度的信息）

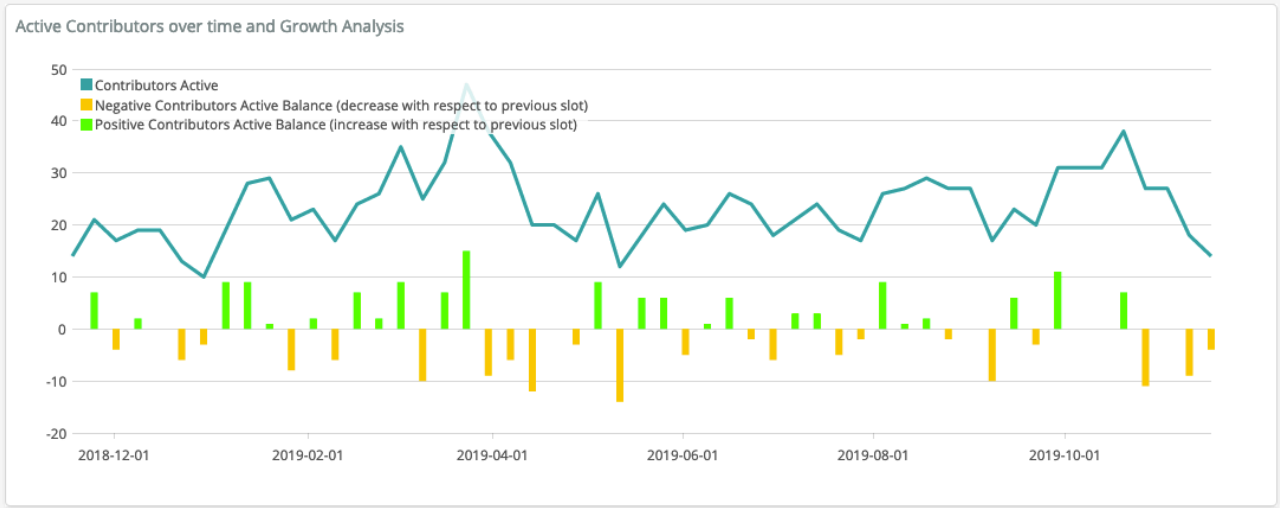
Lines of code added by the top 10 authors

Author	2012	2013	2014	2015	2016	2017	2018
[redacted]	0	133	0	3444	37	12905	1361
[redacted]	0	0	0	0	0	0	59
[redacted]	0	0	0	33	0	0	0
[redacted]	0	0	0	0	0	0	33
[redacted]	0	0	0	0	0	17	0
[redacted]	0	0	0	7	0	0	0
[redacted]	0	0	0	1	0	0	0

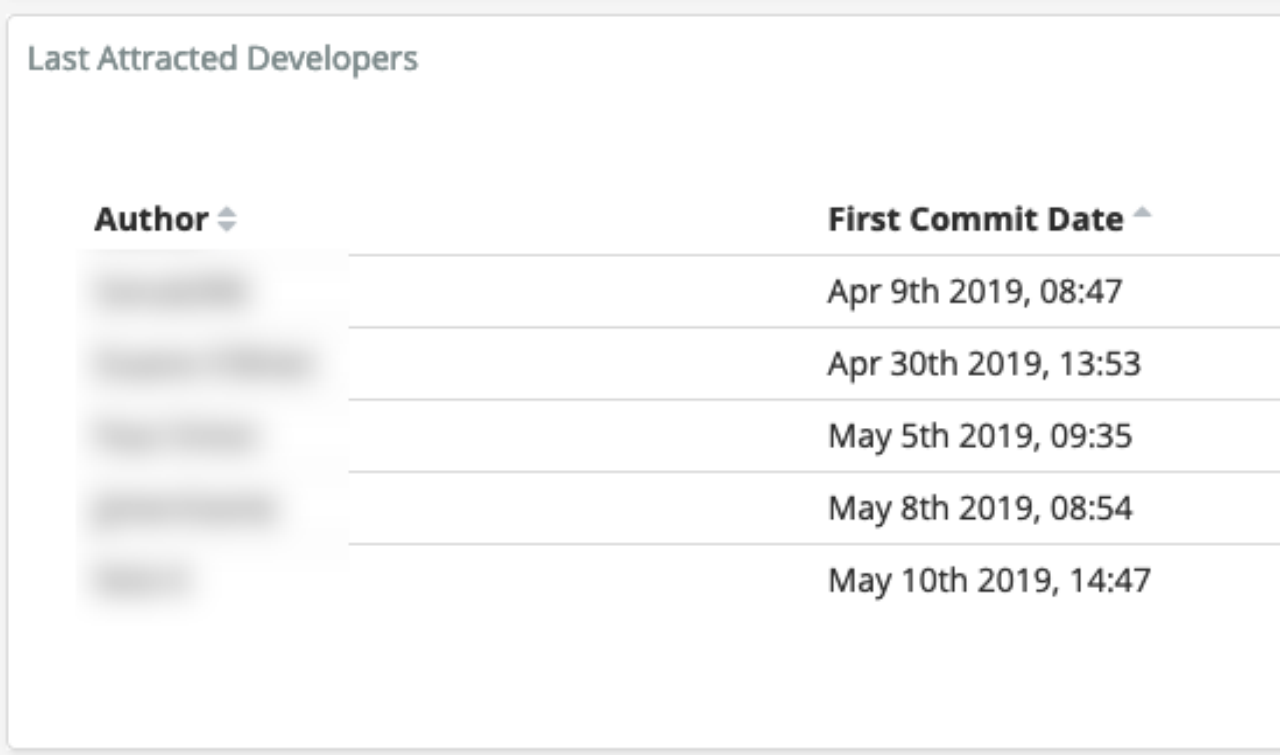
2. 贡献者人数汇总



3. 活跃贡献者数量随时间的变化



1. 新贡献者（按首次贡献日期对贡献者排序）



提供指标的工具

- GrimoireLab
- Augur

数据收集策略

如上所述，部分贡献者信息可以通过 GrimoireLab 和 Augur 等软件获得。然而，有些贡献者洞察不太容易通过跟踪数据获得。在这些情况下，使用社区成员调查或事件注册的方式可以提供所需信息。示例问题包括：

- 采访问题：通常哪些贡献者不会出现在贡献者列表中？
- 采访问题：哪些贡献者经常不被视为重要贡献者，因为其贡献更倾向于 “幕后”？

- 采访问题：您经常与哪些社区成员合作?

此外，社区成员调查可以提供有关项目贡献的更多信息。示例问题包括：

- 李克特量表 [1-x] 项：我正在为项目做贡献
- 矩阵调查项：您在项目中多久参与一次以下活动?
  - 列标题：从不，很少（每月少于一次），有时（每月一次以上），经常（每周一次或多次）
  - 行包括：a) 贡献/审查代码，b) 创建或维护文档，c) 翻译文档，d) 参与项目开发决策，e) 担任社区组织者，f) 指导其他贡献者，g) 亲自参与事件，h) 通过学校或大学计算机项目参与，i) 通过 Outreachy、Google Summer of Code 等计划参与，j) 帮助 ASF 运作（如委员会会议或筹款）

参考资料

贡献者位置

问题：贡献者的位置在哪里？

描述

贡献者所处的地理位置、居住地或工作地。

目标

用于确定贡献者的全球位置，了解他们的工作实践和时区。确定没有贡献的区域，以提高这些区域的参与度；

实现

筛选条件

按以下条件筛选贡献：

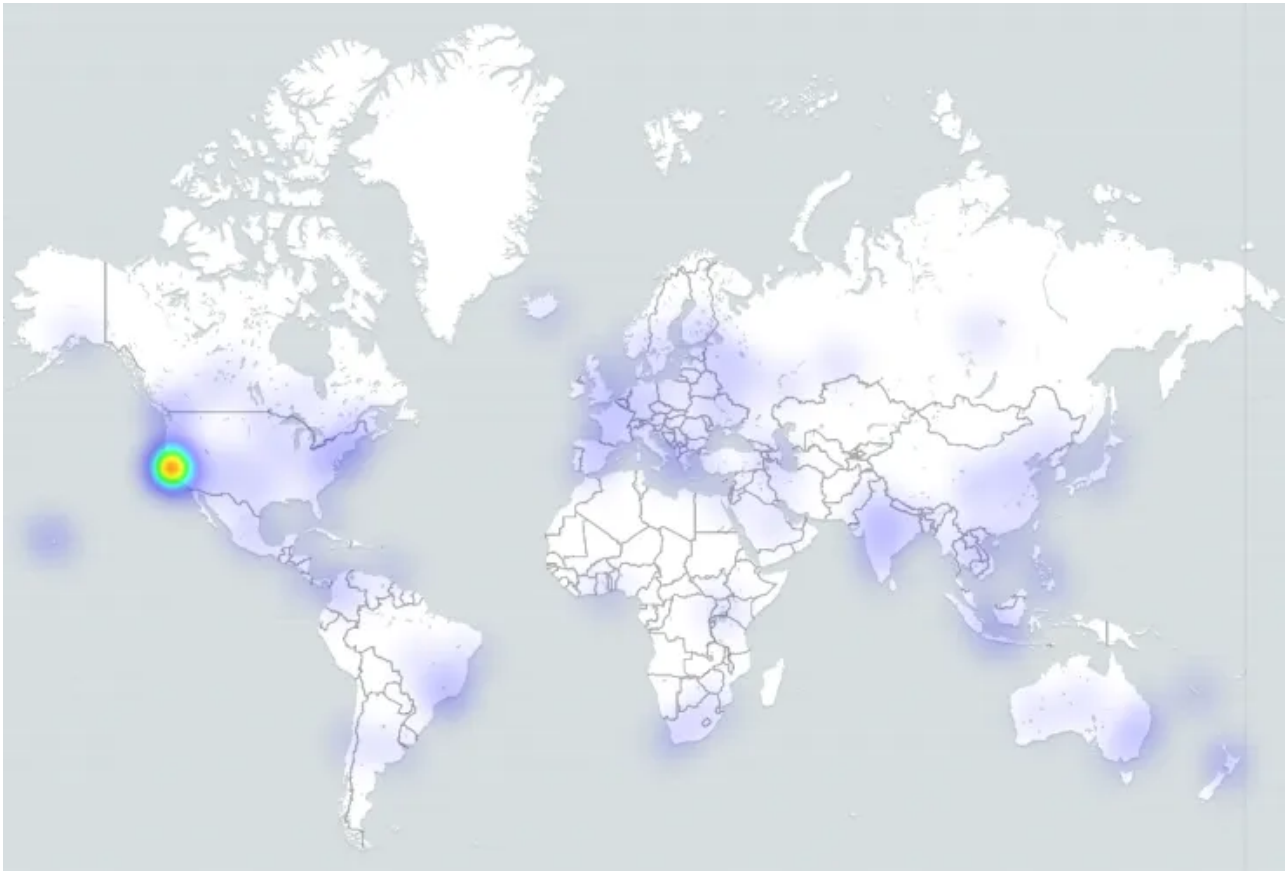
- ** 位置。** 将各区域的位置分组，以进行多级报告。在此语境中，“位置” 是一个故意模糊的术语，可以指地区、国家、州、地域或时区。
- ** 时间段。** 开始日期和完成日期。默认：永久。计算贡献的时期段。
- **贡献者类型**，例如：
  - 仓库作者
  - 议题作者
  - 代码审查 (Code Review) 参与者
  - 邮件列表作者
  - 事件参与者
  - IRC 作者
  - 博客作者
  - 按发布周期
  - 项目的编程语言
  - 项目中的角色或职能

可视化效果

点密度图：

来源：https://chaoss.biterg.io/goto/a62f3584a41c1c4c1af5d04b9809a860

可视热图：



来　　源：https://blog.bitergia.com/2018/11/20/ubers-community-software-development-analytics-for-open-source-offices

提供度量的工具

- GrimoireLab
- Augur

数据收集策略

可以采用不同的方法收集位置信息：

- 在参与度系统中收集贡献者资料的位置信息。
- 使用做出贡献的最频繁位置的 IP 地址地理定位。
- 根据贡献中的时间戳推断地理位置。
- 调查贡献者。

确定贡献者的位置是数据收集的关键挑战。最佳实践是利用参与度系统提供的各种资料信息，如果没有这些信息，可以使用 IP 地理定位来确定该个人最频繁的贡献位置。注意，贡献者可能会在个人资料信息中输入虚假或无意义的位置信息（如 “地球” 或 “互联网”）。注意，IP 地理定位可能会由于 VPN 或其他 IP 屏蔽工具提供大量误报。

另一个考虑因素是外部数据收集工具，如社区调查或事件登记数据，可能交叉引用参与概况的系统。贡献者位置数据可与事件参与者统计信息和演讲者统计信息内联收集。

参考资料

- Gonzalez-Barahona, J. M., Robles, G., Andradas-Izquierdo, R., & Ghosh, R. A. (2008). Geographic origin of libre software developers. *Information Economics and Policy*, *20*(4), 356-363.

组织多元化

问题：什么是贡献的组织多元化?

描述

组织多元化表示一个项目有多少组织参与，以及组织之间不同的参与程度。

目标

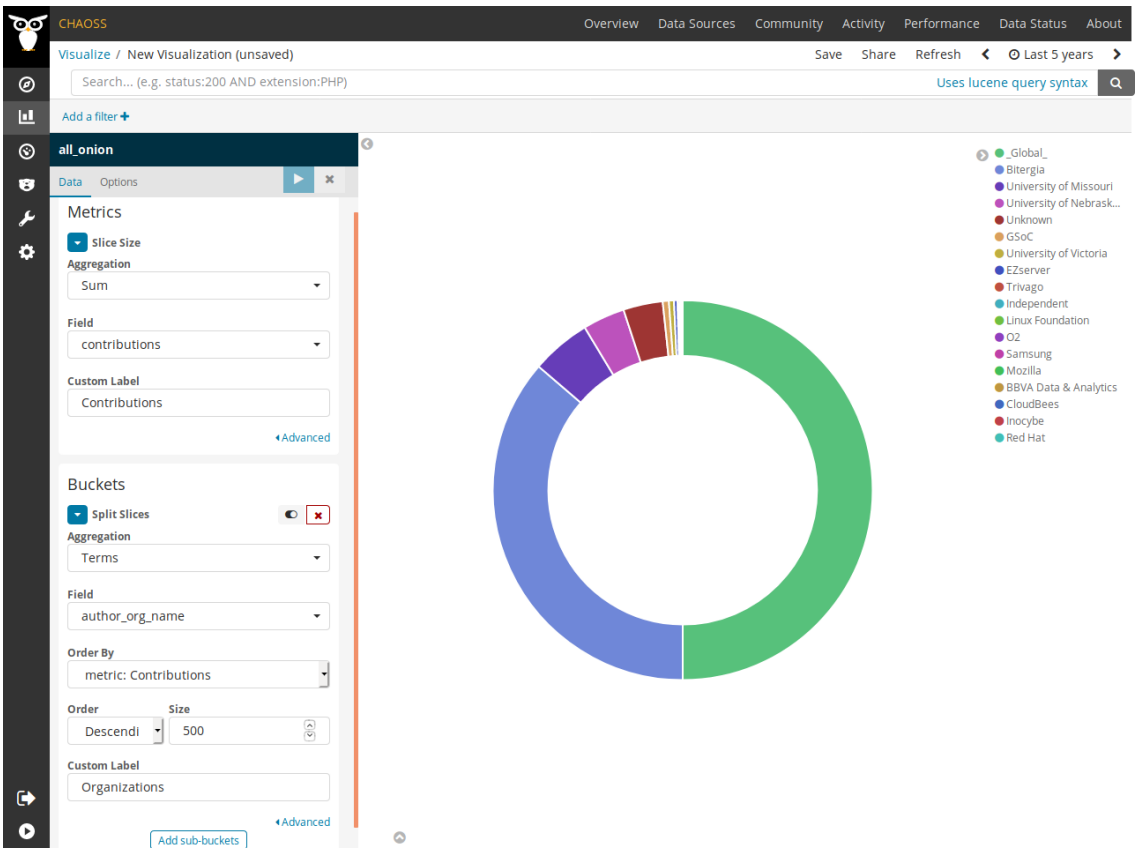
- 获取为项目做出贡献的组织列表。
- 查看各组织在特定时间内的贡献百分比。
- 查看组织结构在特定时间内的变化。
- 获取各组织相关的人员列表。

实现

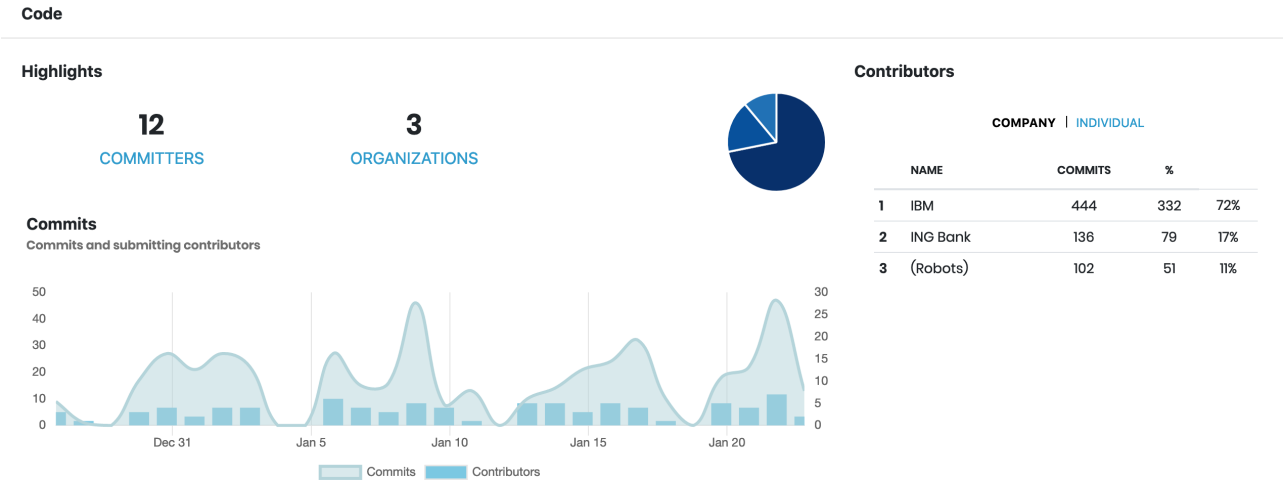
- 从发生贡献的数据源收集数据。
- 确定贡献者的从属关系，对其从属组织作出准确评估。
- 关联贡献相关信息，并将其匹配至适当组织。
- 根据项目需求，您可能需要考虑：如何处理多个电子邮件地址、随时间变化的从属关系，承包商与员工的关系等问题。

提供指标的工具

- GrimoireLab 支持组织多元化指标，开箱即用。GrimoireLab SortingHat 管理身份信息。GrimoireLab Hatstall 用户界面可以修正人员的组织从属关系，甚至记录从属关系的变化。
    - 查看 Bitergia Analytics 的 CHAOSS 实例示例可视化效果。
    - 从 GrimoireLab Sigils 面板集合下载并导入包含此指标可视化效果示例的现成仪表板。
    - 按照说明向任意 GrimoreLab Kibiter 仪表板添加一个示例可视化效果：
        - 新建一个饼状图
            - 选择 `all_onion` 索引
            - 指标切片大小：`Sum` 聚合，`contributions` 字段，`Contributions` 自定义标签
            - 桶拆分切片：`Terms` 聚合，`author_or_name` 字段，`metric: Contributions` 排序依据，`Descending` 排列，`500` 大小，`Organization` 自定义标签
        - 屏幕截图示例



- LF Analytics 在主视图中为提交、提交的问题和通信渠道（目前支持 Slack 和 groups.io）提供组织多元化指标。



数据收集策略

定性

- 组织在项目或生态系统中的足迹
- 组织在项目或生态系统中的影响
- 治理结构中的从属多元化。

定量

- 每个组织的 commits 百分比

- 每个组织的 merges/reviews 百分比
- 每个组织的任意种类的贡献者百分比
- 每个组织贡献的代码行数百分比
- 每个组织提出的问题 (issues) 百分比
- 贡献组织 - 贡献组织的数量是多少?
- 新的贡献组织 - 新的贡献组织的数量是多少?
- 新贡献者组织 - 随着时间推移，对项目作出贡献的新组织。
- 贡献组织的数量 - 在一段时间内参与项目的组织数量。
- 大象系数 - 如果 50% 的社区成员受雇于同一家公司，那就形成了房间里的大象。公式：员工完成 50% 提交次数的最小公司数量
- 从属多元化 - 单个公司的贡献者在所有贡献者中的比例。又称：来自不同公司的维护者。贡献者从属多元化。
- 在具有代码所有权概念的项目中，从属于每个组织的代码所有者的百分比，按其拥有的代码的重要性/大小/LoC 和共同所有者的数量进行权衡。

参考资料

- 潜在实现和参考资料：
  - https://bitergia.gitlab.io/panel-collections/open_source_program_office/organizational-diversity.html
  - Kata Containers 仪表板条目页面（底部）
  - Augur

关注领域 - 地点

目标: 确定贡献发生在哪些物理和虚拟的地方 (例如，GitHub，聊天平台，论坛，会议)

度量指标	问题
协作平台活动	项目使用的跨数字协作平台 (例如 GitHub, GitLab Slack, email) 的活动总数是多少?
活动地点	开源项目活动在何处举办?

协作平台活动

问题: 项目使用的跨数字协作平台 (例如 GitHub, GitLab Slack, email) 的活动总数是多少？ 问题: 项目使用的跨数字协作平台的活动总数是多少?

描述

开源项目使用许多不同的数字通信和协作平台。这些平台可能包括电子邮件、社交媒体、聊天应用程序和代码管理技术 (例如 GitHub, GitLab)。这个度量标准衡量跨协作平台发生的消息活动的位置和数量。

目标

了解社区在哪里协作。当我们去拓展洞察每个项目所遵循的相关流程，如果对交流日志的审查尽可能完整，这些洞察会更准确。展示一个项目是多么的开放和透明。帮助人们找到合适的地方做出贡献，并与项目联系起来。帮助项目维护者确定最佳的渠道数量，可以切实有效的分享信息，同时允许贡献者以最适合他们的方式连接。找到最低障碍的渠道参与。其他相关指标，如: 突发性波动, 项目发展速度, 社会聆听, 活动的日期和时间, 聊天平台包容性

实现

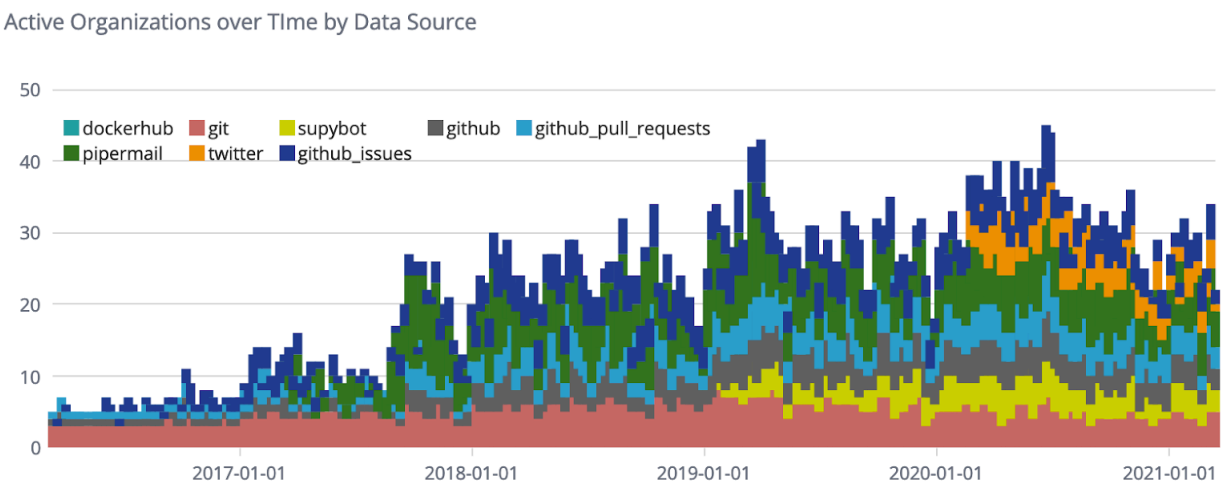
数据收集的单位是平台上的单个活动。与该指标有关的元数据可以包括:

- 时间戳
- 发件人
- 多线程/非线程平台
- 数据收集日期
- 平台消息标识符

过滤条件

- 人数
- 消息数
- 议题和变更请求的评论数
- 平台种类 (邮件列表，irc 等))
- 一周每天的活动量

可视化效果



https://chaoss.biterg.io/app/kibana#/dashboard/ab68fe20-17f2-11e9-872f-e17019e68d6d

提供度量的工具

- Orbit.love 是一个社区管理平台，可以从几个平台获取这些信息: https://docs.orbit.love/docs/adding-activities
- GrimoireLab
- Augur

参考资料

- 相关指标: https://chaoss.community/metric-chat-platform-inclusivity/
- 相关指标: https://chaoss.community/metric-issues-new/

贡献者

- Elizabeth Barron
- Sean Goggins
- Matt Germonprez
- Danial Izquierdo
- Dawn Foster
- Beth Hancock
- Kevin Lumbard

活动地点

问题: 开源项目活动在何处举办?

描述

这个度量有助于了解项目活动在何处举办，从而更好地理解项目吸引全球受众的承诺。项目活动可能包括会议、黑客松、聚会和其他同步聚会。项目活动可能是其它活动的一部分，或与其他活动共同举办。

目标

提供全球范围内项目活动和会面 (meet-up) 地点的信息。

实现

过滤条件

- 线上，线下或者混合
- 时区
- 与另一个会议一起举行
- 地理区域
- 事件类型 (本地聚会 vs. 全球会议)

提供度量的工具

- 项目网站上的共享或公开日历
- 基金会或项目网站上的活动列表

数据收集策略

- 询问事件组织者
- 浏览项目或基金会网站，例如: https://events.linuxfoundation.org/
- 聚合开源事件的站点
- IEEE SA 公开的社区日历: https://saopen.ieee.org/events/

参考资料

贡献者

Elizabeth Barron Dawn Foster Vinod Ahuja Kevin Lumbard Matt Germonprez

Diversity, Equity and Inclusion WG

关注领域	目标
活动多元化	确定活动的多样性和包容性。
治理	确定我们的治理多样化和包容性的情况。
领导力	确定我们的社区领导力有多健康。
项目与社区	确定我们的项目所在的社区的多样化和包容性。

关注领域 - 活动多元化

目标: 确定活动的多样性和包容性。

度量指标	问题
活动行为准则 多元化入场券	活动行为准则如何支持多元化和包容性? 如何使用多元化入场券支持活动的多元化和包容性?
活动多元化 - 欢迎家庭成员参与	让家庭成员共同参与活动如何能够支持活动的多元化和包容性?
活动参与者人口统计特征	一个活动在多大程度上考虑并关注与会者、演讲者和志愿者的人口统计特征?
在活动中的包容性体验	一个活动组织团队在多大程度上致力于提升一个活动的包容性体验?
线上活动的时区包容性	线上活动的组织者如何考虑其他时区的参会者和演讲者?

活动行为准则

问题：活动行为准则如何支持多元化和包容性?

描述

行为准则描述了活动参与者之间的良好行为规则，以及有人违反预期的良好行为时，可以采取哪些途径。具有行为准则的活动表明组织者愿意对问题做出回应。

具有行为准则的活动表明组织者愿意对问题做出回应，这有助于各种背景的人感到包容并舒适地参与活动。活动参与者有权报告问题，并主动帮助减轻存在不良行为的情况。

目标

- 活动组织者要确保提供有效流程来应对行为不端的参与者。
- 活动组织者要确保参与者在活动中获得积极体验。
- 活动参与者要知道如何举报攻击性行为。
- 活动参与者要知道他们在活动中是安全的。

实现

数据收集策略

- 采访和/或调查参与者，进一步了解活动的行为准则符合或不符合其预期的原因。
  - 这次活动对于改进活动的行为准则有什么作用?
  - 哪些示例可以说明活动达到或超过您对行为准则的预期?
  - 参与者在完成登记前是否需要接受行为准则?
- 查看活动网站的行为准则。
- 观察活动中是否发布了行为准则。
- 观察行为准则是否针对活动中的违规行为提供了明确的举报渠道。
- 观察行为准则/活动网站是否向不当行为受害者提供了可能的援助途径，并最终链接到外部机构?
- 浏览活动网站。活动发布行为准则并且针对违规有明确的举报渠道，则符合标准。（注意：理想情况下，行为准则应易于查找）
- 作为参与者或活动工作人员，要观察参与者是否容易找到活动中发布的行为准则。建议在注册网站的突出位置发布行为准则。
- 行为准则相关的调查参与者：
  - 李克特量表 [1-x] 项：该活动在多大程度上满足了您对行为准则的预期。
  - 在注册和活动举办期间，您是否了解行为准则以及违规行为的举报途径? [i]
  - 行为准则的存在是否让您觉得更安全、更有能力充分参与活动? [i]
  - 如果您举报过违规行为，是否得到了满意的处理结果? [i]

参考资料

- 参与者问题处理程序
- 2018 Pycon 行为准则
- 会议防骚扰
- 科技界女性宣言

[i] 部分示例问题重用自 Mozilla 项目。

多元化入场券

问题：如何使用多元化入场券支持活动的多元化和包容性？

描述

邀请不同群体的人可以通过提供特定的门票明确表示出来。这些入场券可带有折扣，进一步鼓励受邀团体成员参加。学生折扣票是一种流行做法。

多元化入场券可以让社区成员参加事件并鼓励新成员加入。失业、就业不足或有其他经济困难的人尤其可能被排除在活动之外。此外，多元化入场券还能够鼓励额外的贡献和正在进行的贡献。

目标

- 证明在活动中增加多元化的有效性。
- 使经济条件有限的人员能够利用折扣参与。
- 鼓励代表性不足的群体成员出席。
- 跟踪外联工作的有效性。

实现

数据收集策略

- 观察网站多元化入场券的可用性和定价。
    - 有多少（不同的）多元化入场券？
    - 获得多元化入场券的标准是什么？
    - 普通入场券和多元化入场券的价格有何不同？
    - 是否鼓励固定参与者赞助多元化入场券？
    - 多元化入场券的赞助者是否列出名字？
    - 网站上是否显示了往期会议所用多元化入场券的数量？
- 采访组织者：
    - 多元化入场券如何分配？
    - 有多少（不同的）多元化入场券？
    - 获得多元化入场券的标准是什么？
    - 有多少参与者使用了多元化入场券？
    - 您在哪里宣传了多元化入场券？
    - 是否有人赞助过多元化入场券？这些人是谁？
- 调查参与者对多元化入场券的看法。
    - 李克特量表 [1-x] 项：学生 [替换为任何受邀群体] 优惠票能有效提高学生 [或其他群体] 的参与度。
    - 是/否项：我知道 [这次会议] 提供了多元化入场券。

    - 是/否/未知项：我有资格获得多元化入场券。
    - 是/否项：一张多元化入场券让我能够参加 [这次会议]。
- 根据多元化入场券跟踪出席人数。
    - 统计不同折扣代码的使用情况来跟踪外联效果，以及哪些群体使用了多元化入场券。要求会议注册系统跟踪折扣代码的使用情况。
    - 在每场事件中统计售出或发放了多少张多元化入场券，并与持有这些入场券的参与者在事件中签到的人数进行比较。

参考资料

- https://diversitytickets.org/
- https://internetfreedomfestival.org/internet-freedom-festival-diversity-inclusion-fund/

活动多元化 - 欢迎家庭成员参与

问题：让家庭成员共同参与活动如何能够支持活动的多元化和包容性?

描述

欢迎家庭成员参与活动可以让参与者携带家人从而降低准入门槛。这可以包括育儿、儿童活动或在活动中针对青少年和家庭成员的主题会议。

目标

- 开源项目要知道一个活动是否对携带家人的参与者具有包容性。
- 活动组织者要知道受邀看护人是否了解可以参与这些家庭导向活动。
- 有 18 岁以下孩子的父母、监护人或看护人要知道他们是否可以带孩子。
- 有 18 岁以下孩子的父母、监护人或看护人必须带着孩子，评估其全家参与的能力。

实现

数据收集策略

- 采访会议人员
  - 问题：会议为需要照顾子女的参与者提供哪些服务?
  - 问题：你们有母婴室吗？如果有，请说明。
  - 问题：活动期间是否提供托儿服务？如果提供，请说明。
  - 问题：如果提供托儿服务，适合什么年龄段的孩子?
  - 问题：是否有既能够照看幼儿还可以看护青少年的活动和照顾?
  - 问题：你们有专门针对儿童的环节吗？如果有，请说明。
- 调查会议参与者
  - 李克特量表 [1-x] 项：活动对家庭成员的欢迎程度如何?
  - 李克特量表 [1-x] 项：任何人都可以带着孩子参与活动，并知道孩子有事情可做。
  - 李克特量表 [1-x] 项：孩子们在会议中有地方玩耍并且不会打扰其他参与者。
- 分析会议网站 [检查表]
  - 会议是否在宣传中表明配备了母婴室?
  - 会议是否宣传了针对儿童和青少年的活动?
  - 会议是否宣传了面向家庭的活动?
  - 会议是否明确邀请参与者携带孩子?
  - 会议是否提供托儿服务，包括青少年空间?

参考资料

- Adacare 会议托儿工具包
- 改善会议托儿服务

活动参与者人口统计特征

Question: 一个活动在多大程度上考虑并关注与会者、演讲者和志愿者的人口统计特征?

描述

与会者和讲演者的人口统计学特征有助于表明活动中不同观点和更广泛的视角和代表性的潜力。因为每个人都有不同的观点，调查每个人口统计群体的反应可以表明，来自不同人口统计群体的人们是否觉得自己的被包容度低于平均值。

目标

- 确定与会者、演讲者和志愿者的多元化，包括性别、年龄、母语和有无残疾。
- 帮助留住来自不同背景的与会者、演讲者和志愿者，为未来的活动做准备。
- 根据主题、会议和路线分组时，确定一组演讲者是否多样化
- 确定新的与会者、演讲者和志愿者是否来自多样化的背景。
- 确定主题和会议选举委员会是否多样化。
- 与他人分享与会者、演讲者和志愿者的人口统计信息
- 建立人口信息的隐私管理声明
- 在人口数据的交叉性方面发现不那么明显的故事

实现

过滤条件

- 人口统计学: 应该分析对主观问题的回答，以收集不同的观点。调查每组人群的回答可以显示出某些人群是否感觉受包容度低于平均水平。
- 主题和会议.
- 会议期间的多样性，不限于单一的系列
- 与会者
- 演讲者
- 志愿者
- 会议委员会

数据收集策略

- 请求与会者和演讲者在会议登记时做人口统计调查
- 在活动之前、期间或之后使用调查来收集与会者和演讲者的人口统计资料 (例如，使用 Open Demographics questions)
- 对于线上活动: 如果线上活动平台支持实时问卷调查，那么使用问卷调查可以更好地了解参加活动的人的背景

参考资料

感谢 *Nikki Stevens* 和 *Marie Nordin* 分享经验和资源，这些经验和资源使得这个度量指标得以创建。

- http://nikkistevens.com/open-demographics/
- https://github.com/chaoss/wg-diversity-inclusion/tree/master/demographic-data

贡献者

- Matt Germonprez
- Justin W. Flory
- Elizabeth Barron
- Matt Cantu
- Georg Link
- Nikki Stevens
- Kevin Lumbard
- Lauren Phipps

在活动中的包容性体验

问题: 一个活动组织团队在多大程度上致力于提升一个活动的包容性体验?

描述

如果森林里的一棵树倒了，没有人听见，那么这棵树真的倒了吗?

同样，我们不仅需要通过组织者的努力来判断一项活动的包容性，还需要通过活动参与者的反应来判断。

当与会者感到被包括在内并受到欢迎时，活动的包容性最具影响力。如果与会者不能在活动上感到被包容和受欢迎，那么组织者为使活动具有包容性而付出的所有努力都是徒劳的。该指标侧重于活动组织者致力于包容性做的工作和活动参与者包容性体验的成果。

目标

该指标的目标是:

- 使活动组织者确保所有与会者都受到欢迎并参与活动
- 鼓励活动组织者在活动中建立包容性，以提供一种归属感
- 在活动中提供度量包容性体验的方法

实现

为与会者提供一个途径，以表达对活动包容性的关注

- 确保任何表达关切的人都能被听到
- 确保违规行为得到处理
- 调查与会者对包容性的看法

数据收集策略

询问与会者和演讲者:

用李克特量表 [1-5](或者表情符号量表) 进行调查:

- 我觉得自己在活动中被包容了
- 我觉得这次活动的工作人员非常重视我
- 在活动过程中，我有机会发出自己的声音
- 我打算再次参加这项活动

采访与会者和演讲者:

- 在这次活动中，你有没有经历过什么让你觉得不受欢迎或者被排除在外的事情?
- 你有没有观察到一些让别人觉得不受欢迎和不被重视的事情?
- 你能提供具体的例子来帮助活动组织者在将来缓解这种情况吗?
- 这次活动的组织者今后在活动包容性方面有哪些可以改进的地方?

参考资料

- Linux 基金会包容性培训
- Fedora 社区徽章

贡献者

- Trisha Rajaram
- Matt Cantu

线上活动的时区包容性

问题: 线上活动的组织者如何考虑其他时区的参会者和演讲者?

描述

全局可访问性在选择活动发生的时间方面具有重要的作用。线上活动的时区包容性确保了活动期间全球社区的参与。通过这一指标，我们能够更加包容世界各地的与会者和演讲者。

目标

在举办线上活动时，应了解全球无障碍环境，以便让所有区域的与会者和发言者都能从活动中受益。这个度量标准的目的是帮助线上活动的组织者制定一个合理的方法，以确保那些在不同时区的人感到被包括在内。

实现

可实施的时区包容性的例子包括:

- 让发言者做一个事先录音，发言者可以在聊天中实时回答问题
- 在特定时区的特定时间段内进行演讲
- 准备并发送一段录音，在一小时内完成直播
- 平台的网络带宽选择，用于现场参与和日后的视频或者音频播放

数据收集策略

用李克特量表1-5对与会者进行调查:

- 这个活动考虑了时区差异
- 这次活动迎合了我对时区差异有关的需要
- 这次活动为我所关心的谈话提供了充分的录音
- 在活动中，我得到了足够的低带宽选项

用李克特量表1-5对演讲者进行调查:

- 为我在活动上的演讲选择的时间块对我来说是公平的
- 我已经或者将要收到一份我自己演讲的录音，以便以后观看

参考资料

媒体文章

贡献者

- Ruth Ikegah
- Trisha Rajaram
- Justin W. Flory
- Matt Germonprez
- Matt Cantu

- Lauren Phipps

关注领域 - 治理

目标: 确定我们的治理多样化和包容性的情况。

度量指标	问题
委员会/理事会多元化	我们的管理委员会或理事会的多元化如何?
项目行为准则	项目行为准则如何支持多元化和包容性?

委员会/理事会多元化

问题：我们的管理委员会或理事会的多元化如何？

描述

管理委员会或理事会由一群领导开源项目的人员组成。这一重要群体的多元化有利于表明项目支持多元化，并且项目中的每个人都有机会发挥领导作用。多元化的委员会或理事会也让不同背景成员的问题更容易被听取和理解。

目标

项目成员需要知道，委员会或理事会中有人代表他们的利益。委员会或理事会要确保避免回声室效应。多元化的委员会或理事会有助于其他人获得归属感并在事业上取得进步。如果与其他领域对比，如技术负责人或社区本身，这一指标有助于委员会了解委员会的当前多元化状态。

实现

数据收集策略

- 观察是否举行了公开选举，以判断候选人是否代表了成员群体。（注意：选举通常偏向来自主流群体的候选人，难以确保选举后会代表少数群体）
- 从项目网页观察委员会或理事会成员的多元化（限制：统计信息可能需要进一步解释）。
- 要求委员会提供委员会或理事会的多元化报告。
- 调查项目成员对委员会或理事会多元化的看法。
  - 李克特量表 [1-x] 项：我感觉自己在委员会或理事会中能够被代表。
  - 李克特量表 [1-x] 项：委员会或理事会关注项目内的少数群体。
- 李克特量表 [1-x] 项：委员会或理事会代表了项目社区的多元化。

参考资料

- https://www.openstack.org/foundation/2018-openstack-foundation-annual-report
- https://www.linuxfoundation.org/about/diversity-inclusiveness/

项目行为准则

问题：项目行为准则如何支持多元化和包容性？

描述

行为准则将可接受的行为传达给项目成员。行为准则还提供针对行为不端者的执行机制。

目标

项目行为准则提供以下内容：

- 了解社区认真对待多元化和包容性。
- 在对项目投入任何时间之前，评估一个社区是否具有多元化和包容性设计。
- 在参与项目之前，确定其人口特征是否受到保护。
- 确保项目伙伴关系和支持者认真对待多元化和包容性，因为这可能影响其自身的声誉和社区健康。
- 了解如何报告和处理违规行为。
- 观察到行为准则正在实行，而不仅仅是具有执行的可能性。

实现

提供指标的工具

- Mozilla 行为准则评估工具

数据收集策略

- 确定行为守则的位置，因其涉及互动和参与项目和事件的主要领域（仓库根、活动入口、通信渠道）。
- 调查项目成员如何看待行为守则对其参与度和安全感的影响。
- 跟踪调查，围绕可发现性、清晰度和相关性进行报告。

定性

- 行为准则通过了 Mozilla 行为准则评估工具
- 采访和/或调查社区成员，进一步了解行为准则符合或不符合其预期的原因。
    - 项目如何改进行为准则？
    - 哪些示例可以说明社区达到或超过您对行为准则的预期？

定量

- 浏览项目网站。发布行为准则并且针对违规有明确的举报渠道，则符合标准。（注意：理想情况下，行为准则应易于查找）
- 行为准则相关的调查参与者：
    - 李克特量表 [1-x] 项：该项目在多大程度上满足了您对行为准则的预期？
    - 李克特量表 [1-x] 项：您对行为准则中描述的社区成员权利和责任有多清楚？

- 您是否了解行为准则以及违规行为的举报途径？[i]
- 行为准则的存在是否让您觉得更安全、更有能力充分参与项目？[i]
- 如果您举报过违规行为，是否得到了满意的处理结果？[i]

参考资料

- CHAOSS 指标：事件行为准则

[i] 部分示例问题重用自 Mozilla 项目。

关注领域 - 领导力

目标: 确定我们的社区领导力有多健康。

度量指标	问题
包容性领导	项目对多元化领导层的支持如何?
导师制	我们的导师计划在支持项目多元化和包容性方面表现如何?
赞助	提供赞助的长期成员在支持社区多元化和包容性方面表现如何?

包容性领导

问题：项目对多元化领导层的支持如何？

描述

领导层是项目和社区文化的核心，因此需要包容的针对性设计和责任界限。“负责人”这个词有时不会引起共鸣；如果项目出现这种情况，就用“有影响力的角色”代替。由于担任领导职务的人员具有较高知名度，因此这一群体的多元化对培养包容性社区至关重要。

目标

向新人发出欢迎的信号，让他们了解如何在项目中取得成功，并给出下文所列的具体限制。

实现

开源领导层示例（又称“有影响力的角色”）：

- 项目维护者（有时称为所有者）
- 具有代码仓合并权限的贡献者
- 作为代码仓组织成员的贡献者
- 开源项目理事会成员
- 官方项目角色（计划代表，技术发言人）
- 项目文件中列出的构建、文件等问题的联系人
- 对项目有 5 个以上拉取请求的人员

步骤 1：了解

第一步是通过提出问题来了解包容性领导的原则。这由 6 条原则组成，可用作检查表。

审查和续期

- 是否所有的领导角色都受时间限制并且到期后需要续期？
- 任期延长是否包括相关社区的审查和反馈机会？
- 人们能否从领导角色体面地转到荣誉退休或顾问角色？

负责人分布

- 负责人在单人可以担任的领导角色数量上是否受到限制？
- 新的领导机会和途径是否由相应领域涉及的社区协商决定？
- 角色要求的标准是否经过相关社区验证？
- 项目是否为角色创建了明确定义？
- 项目是否公开记录角色定义？
- 是否尽可能由团体而非个人承担领导责任？

问责制

- 项目负责人是否同意一个他们可以负责的标准？
- 员工和社区是否知道和理解如何追究负责人的责任？

- 负责人是否意识到其行动代表了组织/项目?
- 负责人是否遵循并持续推动共享的决策框架?
- 领导层是否与项目和社区公开交流不活动或不能活动的时期?

多元化参与

- 项目负责人是否尽可能接受不同的声音和群体?
- 项目是否一贯严格执行行为准则?
- 成为领导层的途径是否明确考虑到包容维度（如时区、语言、带宽、文化规范)?

角色和一致性

- 在整个项目中，每个领导角色是否均受到相似的重视（例如，认可、资源访问权限、机会)?
- 是否所有负责人都明确自己的角色和期望?
- 是否所有负责人都具有共享的基础知识库和技能（如社区参与准则)。
- 项目负责人是否遵循共享的决策框架?
- 项目角色是否明确定义?
- 相关工具是否尽可能一致且连贯?

平等价值

- 项目是否平等地重视和承认不同类型的专家（如技术和非技术)?

承认限制

- 项目负责人是否发现了可能对包容性产生不利影响的脱离控制的技术、资金或其他限制?
- 领导层是否公开明确承认这些限制?
- 负责人是否愿意讨论和实现减少这些限制影响的建议?

步骤 2：评估第二步是根据现有的领导目标评估您的项目/社区。

_ 示例目标：_ 增加通常由项目维护者完成的贡献的百分比（拉取请求审查、响应和尝试复制问题)。

步骤 3：采取行动第三步是根据掌握的情况采取行动。这些步骤取决于您的具体项目社区，本文不提供建议。以下参考含有一些实用信息。

参考资料

- 相关博文
- 创建一个包容性领导团队页面
- 开展开源维护者培训计划
- 为缺失的角色描述、行为准则和其他信息创建拉取请求

背景

上述原则和实践由 Mozilla 员工和志愿者组成的跨职能团队创建和商定，这些人员与主要贡献领域（SUMO、Mozilla Reps、L10N）具有密切联系。

在此处阅读更多信息。

它们已应用于以下项目/产品：

- Mozilla 开源支持计划 (MOSS)
- Mozilla Reps 计划
- Firefox 开发者工具/调试器计划
- 24 个拉取请求（“提交一个包容性错误！”）
- 添加您的项目。

导师制

问题：我们的导师计划在支持项目多元化和包容性方面表现如何?

描述

导师计划是发展和维持社区的重要组成部分，可以邀请新人加入社区，帮助现有成员在社区内成长，从而确保整个社区的健康状态。通过导师计划，我们可以邀请不同的贡献者，为贡献者创造包容的环境，帮助其成长并尽可能回馈社区。

目标

增加新贡献者的数量和多元化，提高每个贡献者的贡献水平。增加贡献者的数量和多元化，鼓励他们成长为新的角色，在社区内承担更大的责任。增加项目的倡导者/传播者人数，以及有偿导师和学员人数。确定和支持导师与学员，培养包容性文化。

实现

数据收集策略

- 采访导师和学员
  - 向社区成员询问现有的正式或非正式指导计划
- 观察人们非正式地指导新贡献者
- 观察学员在指导计划期间和之后做出的贡献
- 在指导计划期间和之后，观察学员在项目中的发展轨迹
  - 学员成为专业领域的专家
  - 学员融入社区
  - 学员愿意向社区中的任何人寻求帮助
  - 学员发展了他们的专业网络
  - 学员承担了社区内部责任
  - 学员在指导项目结束后为社区做出贡献
  - 观察学员保留率与社区惯有保留率的对比
- 观察指导计划的性质
  - 参与指导计划的多种项目（例如，根据 OpenStack 性别报告，有几个项目参与其中）
  - 在整个项目生态系统中针对不同贡献水平（如贡献阶梯、洋葱层）的多种指导计划
  - 指导计划中正式导师的数量
    - 导师经验（指导了多少轮或年）
    - 在接下来的几年中，有多少导师重复担任该职位
- 调查导师和学员
  - 调查李克特项 (1-x)：我个人感觉指导经历很有帮助。

    - 调查李克特项 (1-x)：我建议为此社区提供指导。
    - 导师调查反馈：
        - 您接受过哪些有助于导师活动的培训？
        - 哪些社区支持对您的导师活动有帮助？
        - 您使用什么通信渠道进行指导？
        - 对待新学员，您首先要做的是什么？
- 收集导师和学员的统计信息
    - 完成指导计划的学员人数（假设是有时间/项目限制的指导计划，如 GSoC、Outreachy 或 CommunityBridge）
    - 开始接受指导的学员人数
    - 社区中的导师人数
    - 多元化群体学员人数
    - 学员的地理分布

参考资料

- GSoC 导师指南
- GSoC 学员指南
- Esther Schindler, 2009. Mentoring in Open Source Communities: What Works? What Doesn't?
- OpenStack 性别报告：以指导为重点

赞助

问题：提供赞助的长期成员在支持社区多元化和包容性方面表现如何？

描述

对社区成员的赞助不同于导师制度 (4)。导师提供指导、分享经验并激励学员成长。导师可以来自社区外部。相比之下，赞助人是社区内的一员，用自己的声誉为被赞助人创造在社区内向上发展的机会。

业内最近才开始采用赞助。赞助是增加多元化的有效领导策略。被赞助人在社区中感受到偏见的可能性降低了 90% (1)。被赞助人知道赞助人会在社区中为自己提供支持和争取机会。被赞助人会更愿意加入社区或感到安心，他们知道有人关心自己并且希望认可自己。这可以让被赞助人有效克服过去可能与其多样背景有关的负面经历。赞助人以自己的声誉推动被赞助人在社区中的发展。

目标

- 更长时间保留新成员。
- 扩大贡献者基础，将新的（或不太活跃的）成员转化为更活跃的成员，并最终成为社区负责人
- 培养成员之间更紧密的社区联系。
- 减少社区内部的偏见。
- 为不同背景的新成员提供机会。
- 引导和培训新成员承担新的责任，增强领导力。
- 通过博客、演讲、媒体采访等途径展现为提高多元化和包容性所做的努力，推广赞助形式。
- 找到潜在的具有多元化背景的主题专家作为赞助人。
- 将被赞助人转换为赞助人，让这一循环继续下去。

实现

数据收集策略

- 采访成员：
    - 对于被赞助人：赞助计划在哪些方面让您更加成功？
    - 赞助计划如何改善项目的多元化和包容性？
    - 赞助计划还可以如何改进？
    - 赞助计划是否帮助您在项目中获得更多责任和/或更强领导力？
    - 在活动/会议中收集有关潜在多元化被赞助人的信息
        - 您是否赞助过与自己性别不同的人？
        - 将性别与统计信息维度中的一个相关维度进行交换。
- 调查成员：

- 调查成员："您想过成为其他成员的赞助人吗？"
- 调查被赞助人："现在有赞助人帮助您吗？"
- 李克特量表 [1-x] 项：我正在赞助其他成员。
- 李克特量表 [1-x] 项：我正在赞助与我不同的成员。
- 李克特量表 [1-x] 项：社区中有一个赞助人用声誉为我提供支持。
- 李克特量表 [1-x] 项：赞助计划效果如何？

参考资料

- https://fortune.com/2017/07/13/implicit-bias-perception-costs-companies/
- Sponsor Effect 2.0: Road Maps for Sponsors and Protégés
  - 人才创新中心（付款链接）
  - http://www.talentinnovation.org/publication.cfm?publication=1330
  - The Key Role of Sponsorship:
  - https://inclusion.slac.stanford.edu/sites/inclusion.slac.stanford.edu/files/The_Key_Role_of_a_Sponsorship_for_Diverse_Talent.pdf
- Perceived unfairness is a major reason to leave a community
  - https://www.kaporcenter.org/tech-leavers/
- Sponsors Need to Stop Acting Like Mentors
  - Julia Taylor Kennedy 和 Pooja Jain-Link
  - https://hbr.org/2019/02/sponsors-need-to-stop-acting-like-mentors

关注领域 - 项目与社区

目标: 确定我们的项目所在的社区的多样化和包容性。

度量指标	问题
聊天平台包容性	如何评价社区聊天平台的包容性?
文档的无障碍性	文档如何适应不同的用户?
文档可发现性	用户和贡献者在项目文档中找到他们需要的信息有多容易?
文档可用性	从内容和结构的角度来看，文档的可用性是什么?
议题标签包容性	对于邀请新贡献者、熟练贡献者、非代码贡献者和其他类型贡献者来说，项目议题标注得如何?
项目倦怠	在一个开源项目中，如何识别和管理项目倦怠?
心理安全	社区成员在社区中感到安全的程度，包括参与提交贡献，影响变化，带来他们真实的自我，以及一般性的参与项目?

聊天平台包容性

问题: 如何评价社区聊天平台的包容性?

Description

开源社区需要为贡献者提供交互动的场所。这种交流大部分可能与项目有关，但也可能存在与其他贡献者进行日常的社交空间。该指标描述了聊天平台中的同步通信，其中包括公共聊天室、实时对话和视频会议平台。

聊天平台对于一个正常运作的开源社区来说是必不可少的。聊天平台通过支持不同技能领域的跨团队协作来支持一个健康的项目，帮助同行协作者更快地找到对方并与之互动，提高项目透明度，使团队成员能够提前发现潜在问题，并通过跨越组织层次结构节省时间。能够保留聊天记录的聊天平台也成为新手熟悉项目历史、实践和文化的参考。

聊天平台应该被透明的选择、管理，使用包容性的语言，以加快创新性的解决问题和维持贡献者的参与。

目标

- 帮助决策者选择聊天平台。
- 提供贡献者和潜在贡献者可以用来评估任何特定项目的具体背景的想法。
- 确定聊天平台的特有功能，如在过滤器下识别的功能，表明具有更大的包容性。

实现

筛选条件

这些过滤器如何度量包容性？把它想象成一个办公室。有多少个团队 (聊天平台)？他们有多活跃？例如，非常活跃的聊天可能很难跟上，而且这种难度因人而异。

- 通用:
  - 活跃聊天室/群组数目
  - 每天、每周、每月的消息数
  - 每天、每周、每月活跃参与者人数
  - 被动参与者的数量 (即阅读但不参与)
  - 外部参与者的数量 (即那些与致力于开源项目的领先公司没有关系的人，例如 Red Hat 和 Fedora 项目)
- 身份:
  - Number of (independent) volunteers
  - 企业志愿者人数 (由另一个实体支付给志愿者工作的人数)
  - 雇工人数 (雇员或外包)
- 没有持久连接的平台 (即 IRC) :
  - 每天连接到房间的次数
  - 重复登录次数

可视化效果

- 社区特定的聊天平台最活跃的时间
- 活跃的新聊天成员数量与老聊天成员数量的比较
- 词云 (用于公共聊天平台，例如，Mattermost rooms)
- “On-topic” vs. “off-topic” 话题中的” 与 “话题外的” 对话
    - 某些关键词或短语出现的频率等等

提供度量的工具

GrimoireLab Perceval 目前支持从各种聊天平台收集数据。

数据收集策略

- **从聊天平台跟踪数据**:
    - 参见支持的数据收集平台 in Perceval.
- **聊天平台的开源状态**:
    - 参见 CHAOSS blog post on open source platform status。
    - 聊天平台是否提供开源客户端应用程序?
    - 聊天平台是否提供开源服务器的实现?
    - 聊天平台是否提供了集成数据收集或桥接的 Open API?
- **低带宽连接的特性**:
    - 是否需要使用持久的 Internet 连接 (例如 IRC)?
    - 聊天平台是否以最小的带宽运行?
- **用于社区管理和审核的功能**:
    - 聊天平台是否为版主提供强大的审核工具?
    - 聊天平台是否有机器人功能，或者是否需要完全人工干预来进行审核?
- **国际化的特性**:
    - 聊天平台是否可在多个国家使用，还是有地理限制?
    - 聊天平台是否提供翻译工具?
- **访问功能**:
    - 聊天平台是否支持可访问性 (例如，对屏幕阅读器友好)?
    - 聊天平台对于新手来说是否易于使用?
    - 聊天平台是否适用于各种设备?
- **隐私功能**:
    - 聊天平台是否要求用户验证个人信息或提交个人信息以供注册?

参考资料

1. 喜欢还是讨厌聊天? 4 个远程团队的最佳实践 — *Opensource.com*
2. 聊天平台的开源状态 — *Chaoss.community*
3. **审核建议** (可通知选择平台):
   (a) 包容社区的审核指南 — *rhea.dev*
   (b) 参与 & 审核指南 — *drupaldiversity.com*

文档的无障碍性

问题: 文档如何适应不同的用户?

描述

由于文档在开源项目中所起的作用，文档的无障碍性至关重要。文档用户具有不同的能力，要求以不同的格式提供文档，以便为这些不同的用户提供同等的授权。其目标是为了促进项目最广泛的贡献者的理解能力。

目标

这些目标有助于衡量您的文档是否为广大受众所接受，同时又不会对其预期受众的部分群体造成不成比例的人为债务。

- **无障碍屏幕阅读器** — 文档可根据屏幕阅读软件的标准访问.
- **学习灵活性** — 有不同认知方式、感觉差异和神经多样性的人都可以使用文件。(1).
- **盲人或视障人士** — 文档对主要阅读文字的人来说是无障碍的。图表和图像是非无障碍类型的文件的例子。

实现

注意: 注意文档的目标读者。由于项目和可能做出贡献的人的范围很广，文件应满足所有受众的不同要求。

数据收集策略

* 使用**工具**来评估屏幕阅读器的友好性 (5) (6) 来确定文档是否对屏幕阅读器友好?

- **采访**新人可以找出文档是如何帮助，(a) 理解贡献过程，和/或，(b) 帮助完成项目中的任务
  采访问题举例:
  - 您通过使用文档理解贡献过程的体验是什么?
  - 当您有关于在项目中工作的问题时，您查阅文档的体验是什么?
  - 您对文档出现的技术术语感到满意吗?
- **调查**项目成员
  - 矩阵项: 当您需要查找有关此项目的过程、政策或指南的相关信息时，以下哪项描述了您的经验? (2)
    - 矩阵行可以是:
      - 邮件列表交流
      - 聊天交流
      - 执行代码审查
      - 代码被采纳的流程
      - 行为守则
      - 新入职人员

        - 许可证
        - 商标
        - 添加新的提交者或者贡献者
        - 项目发布
        - 投票流程
        - 安装过程
        - 功能开发
        - API 集成
        - 测试套件
        - 架构总览
        - 用户指南
      - 矩阵列可以使:
        - 总是很容易找到
        - 很容易找到
        - 很难找到
        - 很难找到
    - 多项选择: 当您开始参与项目时，是否遇到过与文档无障碍性相关的任何挑战 (例如，语言障碍、文档的可发现性、文档结构)？？(2)
      - 回答选项:
        - 没有挑战
        - 一些挑战
        - 几个挑战
        - 许多挑战
      - 开放式的后续问题，如果答案不是 “没有挑战”: 描述一个例子，当你经历挑战，该挑战如何影响你，以及如何，如果有，你克服了挑战。
    - 开放式问题: 对于改进项目的政策、流程或提供给新贡献者的指南，你有什么建议?? (2)
- 寻找关于可读性和可扫描性的组织结构，例如:
    - 标题
    - 文本和代码块
    - 标号或者段落
    - 锚点
- 通过考虑以下因素来评估搜索能力:
    - 用户能多容易地找到这个文档?

    - 用户在文档中找到他们需要的东西有多容易?
    - 使用键盘操作文档容易吗?
- 提供一个快速的微观调查，只有一个问题给文档的读者 (例如，当离开文档页面时，底部页或弹出):
    - 是/否的问题: 这个文档页面对你来是否无障碍性?
    - 李克特量表 [1-x] : 这个文档对你来说有多容易理解?
    - 简短回答: 您对文档的无障碍性有什么看法?
- 与文件的预期用户进行**预览**文档。观察他们如何交互和使用文档，以及他们在哪里卡住了。这可以是一个视频会议会话，文档的用户在其中共享他们的屏幕。
- 要求文档的用户写一个**摩擦日志**，描述他们在文档中遇到的问题。这为文档编辑提供了具体的使用案例，以了解如何为特定用户改进文档。
- 考虑是否为不同的受众提供**不同版本的文档**? 例如，一个简要的版本和一个非常详细的版本。

参考资料

1. 为在 Kubernetes 做贡献的神经分化症患者打破障碍
2. Apache 基金会社区调查问卷 2020
3. GNOME 无障碍报障团队
4. W3C Web 内容无障碍访问指南
5. W3C Web 无障碍性评估工具
6. 一些用于检测 web 无障碍性的快速测试方法
7. Knowability - 一个专门从事无障碍性建设的团体
8. 关于无障碍性度量指标的思考
9. Paypal 无障碍指南清单
10. 通用设计

文档可发现性

问题: 用户和贡献者在项目文档中找到他们需要的信息有多容易?

描述

文档的可发现性至关重要，因为它在开源项目的包容性中扮演着重要角色。如果人们不能轻松地找到并浏览文档主体，或者找到他们所寻找的答案，他们就不太可能使用或为项目作出贡献，从而降低了项目吸引不同参与者的能力。文档可发现性的目标是确保可搜索性和可见性，以及提高不同受众的包容性。

目标

-**可见性** — 文件很容易找到和参考。-**可搜索性** — 用户可以根据自己的能力轻松地浏览文档，并且可以通过搜索找到自己需要的相关信息。-**发现方法** — 确定人们发现文档的方式。-**包容性** - 知识在项目贡献者之间更公平地分享，新的贡献者更容易参与。

实现

数据收集策略

调查项目成员李克特量表 (很容易找到——不可能找到) 关于文档的可发现性，比如:

- 邮件列表存档通信
- 邮件列表会员管理
- 聊天平台存档通信
- 执行代码审查
- 代码合入流程
- 行为准则
- 新员工培训
- 许可证，商标
- 项目领导
- 项目发布
- 投票流程

多选复选框: 如何发现项目文档?

- 网站
- 代码仓
- 软件中的文档
- 搜索引擎
- 其他

多项选择: 当您开始参与项目时，是否遇到过与文档的可发现性相关的任何挑战 (例如，语言障碍或文档结构)？ 答案选项:

- 没有挑战

- 一些挑战
- 几个挑战
- 许多挑战

如果答案不是 “没有挑战”，那开放式的后续问题是:

- 举个例子，说明你经历了什么挑战，什么时候经历的，挑战如何影响你，如果有的话，你是如何克服挑战的。
- 开放式的后续问题:
- 您对于改进项目文档中关于新贡献者可用的政策、流程或指南有什么建议?
- 与文件的预期用户进行预览文档。观察他们如何交互和使用文档，以及他们在哪里卡住了。这可以是一个视频会议会话，文档的用户在其中共享他们的屏幕。
- 要求文档的用户写一个**摩擦日志**，描述他们在文档中遇到的问题。这为文档编辑提供了具体的使用案例，以了解如何为特定用户改进文档。

参考资料

- 知识的诅咒
- 知识库 101
- 在开源项目中优先处理文档的 5 个技巧
- 为文档站点构建索引: 设计中的 5 个最佳实践

文档可用性

问题：从内容和结构的角度来看，文档的可用性是什么？

描述

文档可用性确保不同用户都能理解开源项目文档的关键作用。文档可用性包含结构、内容、术语和可读性等问题，目的是最大限度推动贡献者对项目的理解。

目标

- **技术术语** - 文档使用适当程度的技术术语并提供必要的术语解释，确保特定项目的入门贡献者能够理解文档。
- **结构清晰度** - 文档易于阅读和理解。
- **可读性** - 文档语言应简洁明了，使用常用词和短句，确保非母语者或可能不了解类似缩写惯例或推论模式的人能够理解。
- **语言包容性** - 文档应避免使用非包容性语言（例如 “brogrammer” 语言或排斥性/贬义语言）。
- **语言多元化** - 针对目标受众和不同语言提供通用语言的文档。
- **时间/注意力多元化** - 文档包容性让人们可以在不同时间读取或设置/运行命令。还应考虑到需要照顾孩子或具有其他照护职责的人员，这些人员随时可能将注意力转移到屏幕之外。

实现

数据收集策略

- 采访新人，确定文档如何帮助贡献者 (a) 理解贡献过程，和/或 (b) 完成任务。样本采访问题：
  - 描述一下您是如何使用文档了解贡献进程的。
  - 描述一下您在社区工作中遇到问题时是如何使用文档的。
  - 描述您是如何使用文件了解和帮助外联工作的。
  - 您对这些技术术语感到满意吗？（适用于李克特量表 [1-5] 的调查）
  - 有您不理解的术语或语言吗？
  - 您对改进项目的政策、流程或新贡献者可用的准则有什么建议？
  - 采访后，社区可追踪回复，记为 `Positive experience` 、`Negative experience` 或 `Neutral experience` ，逐月报告，了解随时间推移的改进情况。
- 提出有关可读性和可扫描性的问题，如：文档是否使用组织结构，如：
  - 标题
  - 文本和代码块
  - 项目符号与段落
  - 锚点

- 与文档的预期用户进行预审。观察他们如何使用文档以及在哪里遇到问题。可以采用视频会议会话，让文档用户共享屏幕。
- 要求文档用户编写摩擦日志并描述其在文档上遇到的问题。这为文档编辑提供了具体用例，以了解如何针对具体用户改进文档。
- 考虑为不同受众提供不同版本的文档。例如，轻量版本和非常详细的版本。

资源

- W3C Web 内容无障碍指南
- W3C Web 无障碍评估工具列表
- 一些评估 Web 辅助功能的快速测试
- 专门研究辅助功能的小组
- 关于辅助功能指标的思考
- GNOME 与辅助功能
- Paypal 无障碍指南列表
- 核心基础架构倡议：最佳实践徽章
- 打破神经多样性个体的 Kubernetes 贡献障碍
- `文档基础`
- `文档接口`
- 摩擦日志
- 斯坦福：屏幕阅读器测试

议题标签包容性

问题：对于邀请新贡献者、熟练贡献者、非代码贡献者和其他类型贡献者来说，项目议题标注得如何？

描述

议题标签包容性有助于衡量不同熟悉程度（如新人、偶尔贡献者和核心贡献者）、不同技能（如语言、框架、API、文档、前端和后端）、不同目标（如代码和非代码贡献）的贡献者对议题的承受能力。

目标

新人友好：议题包括适当标注的议题（例如，“对新人友好的第一个提议”，“新人友好”），供新贡献者入门

导师支持：议题确定一位导师，为特定议题的审查过程提供指导和帮助（例如，“可提供导师”标签）

议题列表多元化：议题提供了与不同类型贡献（代码和/或非代码）有关的多元化标签列表（例如，“文件”标签）

可用标题和描述：议题标题和描述遵循文档可用性指标目标

标签的一致使用：议题一致使用带有明显颜色的标签特定列表。例如，每个标记系列都有不同的颜色：

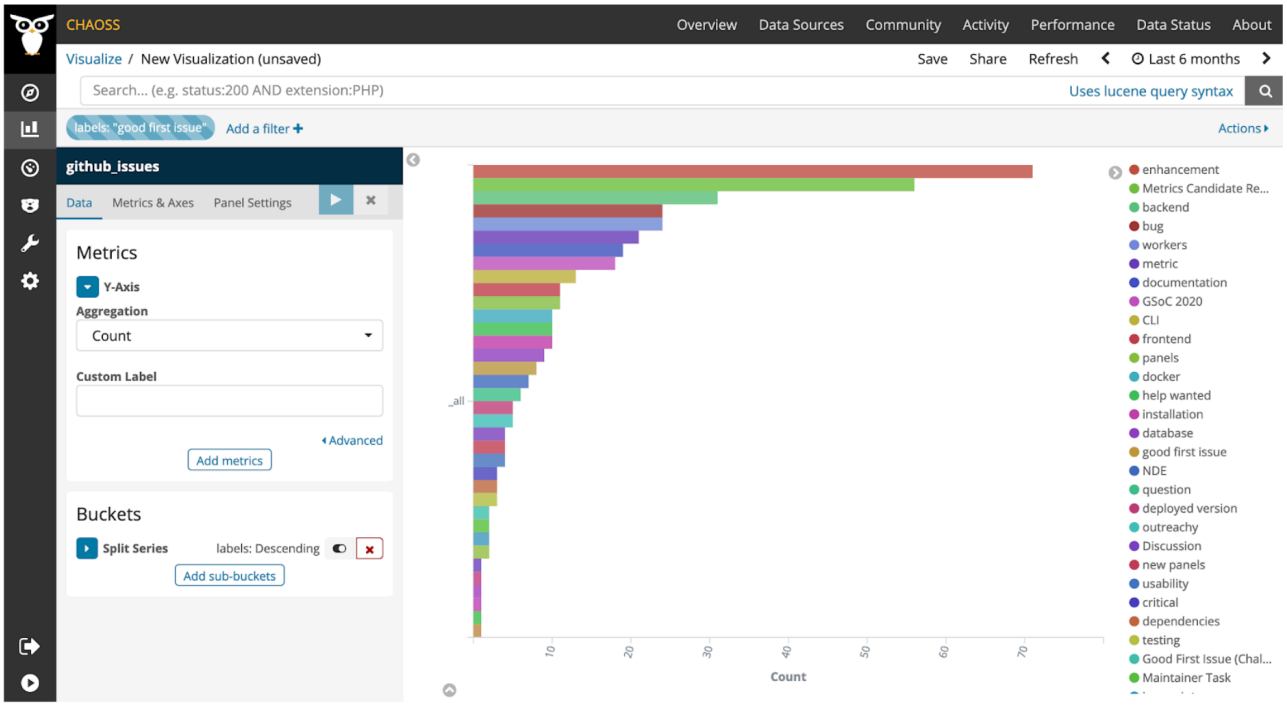
- 议题类型：“功能”与“缺陷”与“文档”...
- 议题技能：解决议题所需的技能（例如 js、html、css）
- 议题熟悉程度：提及所需的最低熟悉程度（“适合新人”或“偶尔贡献者”或“核心贡献者”...）

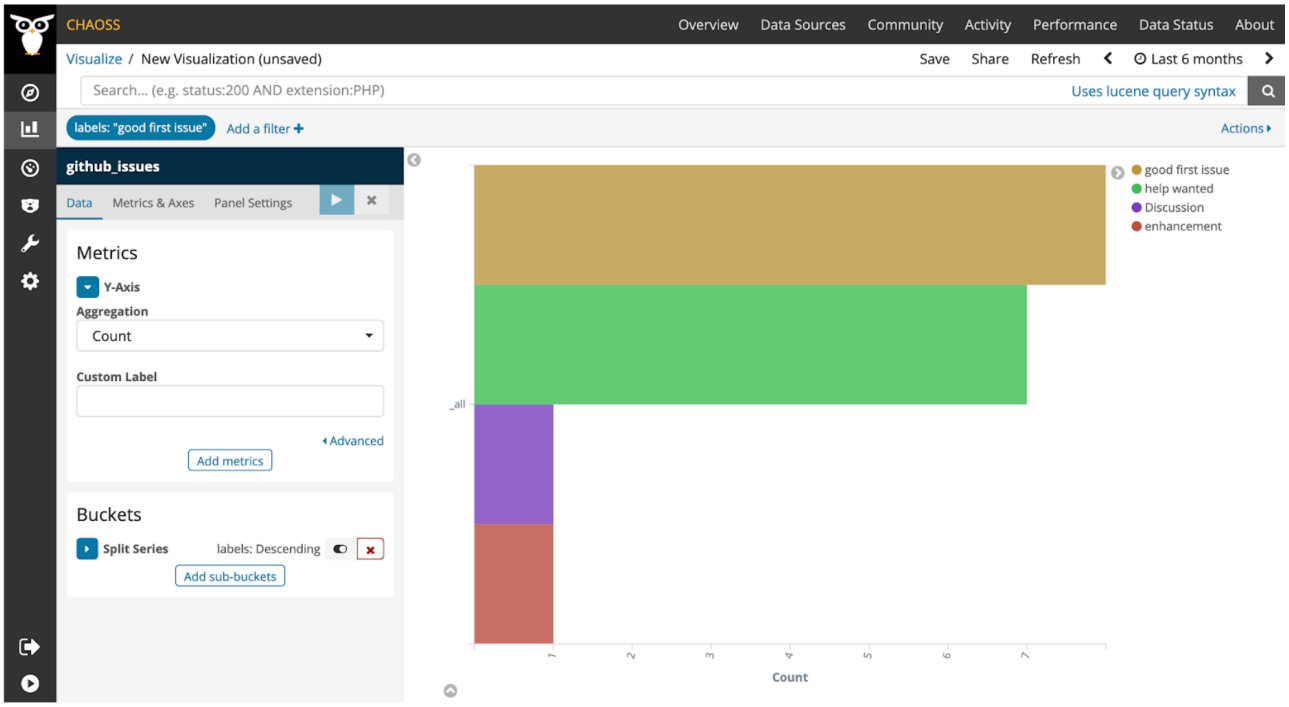
实现

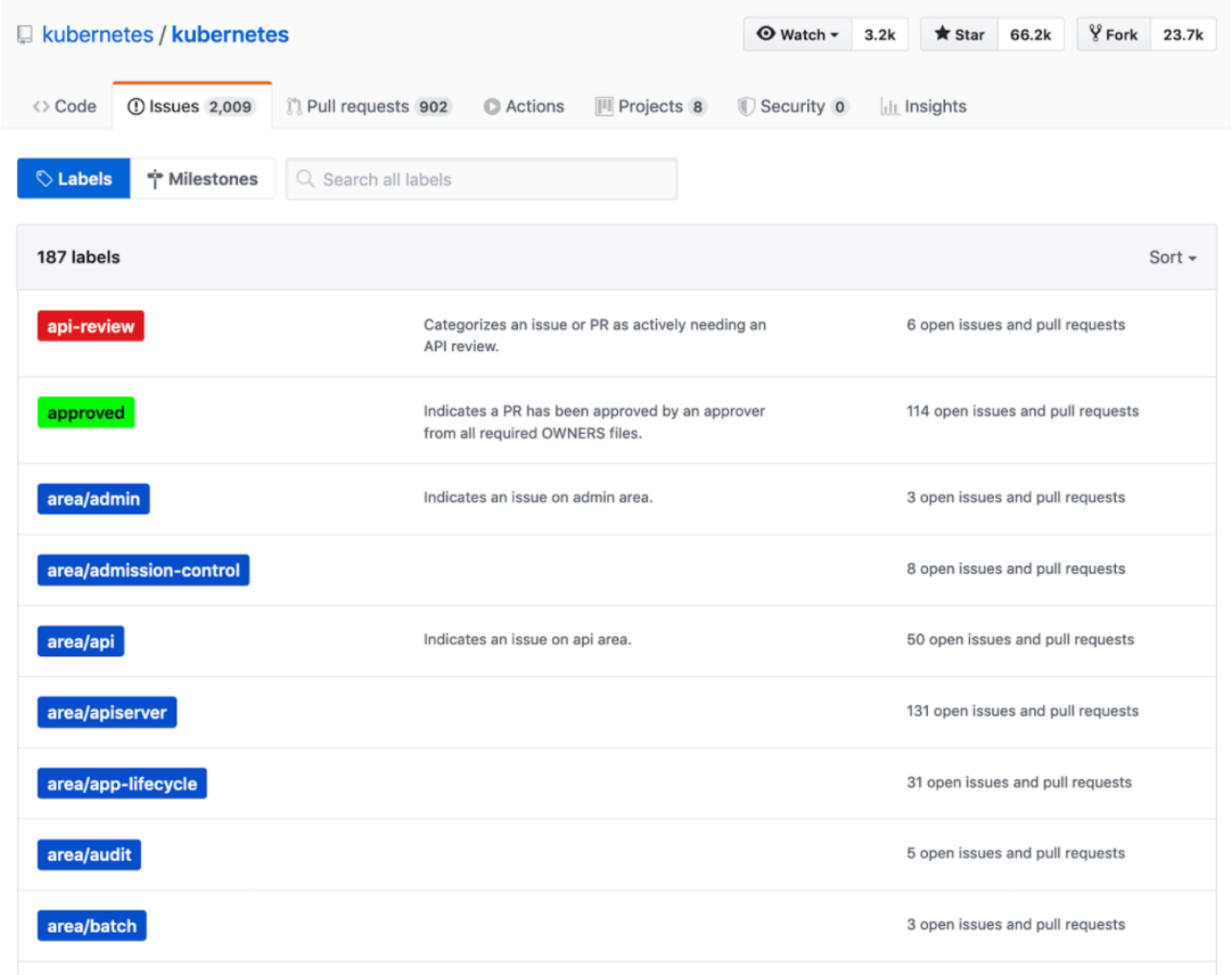
筛选条件

- 标签类型
- 仓库类型
- 待解决议题的时长
- 待解决议题的数量
- 创建议题的日期
- 代码相关议题与文档相关议题

可视化效果


CHAOSS
Overview
Data Sources
Community
Activity
Performance
Data Status
About
Visualize / New Visualization (unsaved)
Save
Share
Refresh
Last 6 months
Search... (e.g. status:200 AND extension:PHP)
Uses lucene query syntax
labels: "good first issue"
Add a filter +
Actions
github_issues
Data
Metrics & Axes
Panel Settings
Metrics
Y-Axis
Aggregation
Count
Custom Label
Advanced
Add metrics
Buckets
Split Series
labels: Descending
Add sub-buckets
_all
10
20
30
40
50
60
70
Count
enhancement
Metrics Candidate Re...
backend
bug
workers
metric
documentation
GSoC 2020
CLI
frontend
panels
docker
help wanted
installation
database
good first issue
NDE
question
deployed version
outreachy
Discussion
new panels
usability
critical
dependencies
testing
Good First Issue (Chal...
Maintainer Task


CHAOSS
Overview
Data Sources
Community
Activity
Performance
Data Status
About
Visualize / New Visualization (unsaved)
Save
Share
Refresh
Last 6 months
Search... (e.g. status:200 AND extension:PHP)
Uses lucene query syntax
labels: "good first issue"
Add a filter +
Actions
github_issues
Data
Metrics & Axes
Panel Settings
Metrics
Y-Axis
Aggregation
Count
Custom Label
Advanced
Add metrics
Buckets
Split Series
labels: Descending
Add sub-buckets
_all
1
2
3
4
5
6
7
Count
good first issue
help wanted
Discussion
enhancement



来自： https://github.com/kubernetes/kubernetes/labels

提供指标的工具:

- Grimoire Lab
- Augur

数据收集策略

- 为每个项目确定所用议题标签的发布清单
  - 一般标签：
    - 在项目的标签列表和议题列表（使用中的标签）中，存在识别 “特性”、“缺陷” 和 “文档” 标签、“前端” 和 “后端” 标签的一般需求的标签
  - 包容性标签：
    - 新人友好型会在项目的标签列表和议题列表（使用中的标签）中查找（新人、第一次）
    - 导师友好型会在项目的标签列表和议题列表（使用中的标签）中查找（导师）
  - 技能标签：
    - 在项目的标签列表和议题列表（使用中的标签）中，存在识别所需技能的标签（如 Java、Python、HTML、机器学习）

- 观察项目中各个议题所用每个标签的频率

参考资料

- GitHub Satellite

项目倦怠

问题: 在一个开源项目中，如何识别和管理项目倦怠?

描述

倦怠是一种由于长期压力而导致的精神、情感和身体上的疲惫状态。工作倦怠的三个维度是能量耗竭或衰竭、玩世不恭或与工作心理距离增加以及职业效能感降低 (WHO，2020)。项目倦怠可能发生:

- 当项目成员被项目参与所预期的工作量压得喘不过气来时
- 作为社区和贡献者活动的结果，包括技术工作、社区管理和组织工作。

目标

项目倦怠应在项目内部加以理解和有效管理，以改善所有人的幸福感。这个指标的目的是:

- 与社区一起制定策略，以帮助项目相关的工作倦怠
- 确定引起项目倦怠的主要活动
- 通过在倦怠发生之前提前将其识别出来，来帮助人们管理倦怠。
- 帮助人们在项目活动中做出维持健康生活的决定。

实现

数据收集策略

- 以下是一种通过一组有意义的问题来更好地评估开源项目贡献者和维护者的幸福感的方法。
    - https://cdn.ymaws.com/www.palibraries.org/resource/collection/9E7F69CE-5257-4353-B71B-905854B5FA6B/Self-CareBurnoutSelf-Test.pdf
    - 跟踪这些问题，聚合社区的结果，并使用个人分数，如果他们也是共享的。
- 调查: 询问项目中个人的幸福感
    - 调查李克特条目 (1-x): 我觉得在这个开源项目上工作充满活力。
    - 调查李克特条目 (1-x): 我在这个项目的工作有疲惫不堪的感觉。
    - 调查李克特条目 (1-x): 当我在这个开源项目工作时感到疲惫。
    - 调查李克特条目 (1-x): 当我在这个开源项目工作时，我花时间照顾自己，自发的锻炼身体，以增强健康和积极的幸福感。
    - 调查李克特条目 (1-x): 在过去的六个月里，我考虑过离开这个项目。
    - 调查李克特条目 (1-x): 由于项目相关的压力，我已经考虑或者已经从项目中休息了一段时间。
    - 调查李克特条目 (1-x): 我已经考虑或已经从项目中休息，以平衡我生活的其他部分。
    - 调查李克特条目 (1-x): 我似乎从来没有足够的时间来完成这个项目的所有事情。

    - 调查李克特条目 (1-x): 我不得不忽略一些与这个项目相关的任务，因为我有太多的事情要做。
    - 调查李克特条目 (1-x): 我觉得我在项目中的贡献对我来说是有价值和回报的。
    - 调查李克特条目 (1-x): 我觉得我的声音在项目中有被听到。
    - 调查李克特条目 (1-x): 我们可以在项目中公开谈论我们是如何做的，感觉如何和检查彼此。
- 跟踪数据: 探索在线数据，以便更好地了解项目中个人的幸福感
    - 根据每个人随时间的贡献数量来分析活动指标，看看在延长贡献时间之后是否会突然下降。
    - 分析是否有长期的连续贡献突然终止。
    - 分析是否有一小部分人在做大量贡献 (见巴士系数或大象系数 度量指标)
- 访谈: 与开源项目贡献者和维护者谈论他们自己对相关调查条目的理解
- 贡献者问题:
    - 你觉得在这个项目中工作得怎样?
    - 糟糕的幸福状态是否影响了您参与这个开源项目? 如何影响?
- 维护者问题:
    - 我们应该如何监测个人的幸福感?
    - 在开源项目中，如何衡量社区成员的幸福感?
    - 如何判断项目贡献者的幸福感?

参考资料

1. 关于开源社区中的倦怠你需要知道什么
2. 一个开源维护者在精疲力竭之后会做什么?
3. Raman, N., Cao, M., Tsvetkov, Y., Kästner, C., & Vasilescu, B. (2020). 开放源码中的压力和倦怠: 走向发现、理解和减轻不健康的互动。在国际软件工程会议上，新思想和新成果 (ICSE-NIER)
4. 2020 年领导力报告中的情绪适应力 (“但不是针对开源软件”)
5. 帮助人们确定他们处于哪个倦怠阶段，或者他们是否处于一个更严重的倦怠阶段: 避免过度疲劳和过上幸福生活的实用指南
6. 倦怠
7. 我是如何从困难中了解倦怠的

心理安全

问题: 社区成员在社区中感到安全的程度，包括参与提交贡献，影响变化，带来他们真实的自我，以及一般性的参与项目?

描述

心理安全是实现社区包容性目标不可或缺的一个关键因素。根据创造性领导力中心的说法，心理安全可以分为几个不同的阶段:

- 阶段 1 – 包容性安全: 包容安全满足了人类连接和归属的基本需求。在这个阶段，你觉得做你自己是安全的，并被接受你是谁，包括你独特的属性和定型特征。
- 阶段 2 – 学习者安全: 学习者安全满足学习和成长的需要。在这个阶段，通过提问，给予和接受反馈，实验和犯错误，你感到在学习过程中交流是安全的。
- 阶段 3 – 贡献者安全: 贡献者安全满足了产生差异的需要。你觉得用你的技能和能力做出有意义的贡献是安全的。
- 阶段 4 – 挑战者安全: 挑战者安全满足了使事情变得更好的需要。当你认为有机会改变或提高的时候，你会觉得勇敢地说出来并挑战现状是安全的。

在心理安全程度较高的社区，贡献者更有可能提出变革建议，并承担领导和/或决策角色。社区心理安全影响多样性、公平性和包容性，通过以下指标表现出来:

- 多样性: 来自代表性不足群体的贡献者更有可能成为积极的参与者，并鼓励其他人也这样做。
- 平等: 决策和领导作用分布均匀，包括传统上被边缘化的群体。
- 包容性: 那些来自边缘化或代表性不足背景的人感到受欢迎，就像他们能够在一个安全的环境中将真实的自我带入项目。
- 吸引力: 创造一个安全的环境可以增加吸引新贡献者的可能性。
- 留存: 如果贡献者觉得这样做安全，他们更有可能保持与项目的联系

如果符合以下条件，将对贡献者的多样性、包容性和归属感产生积极影响，并最终留住贡献者:

- 社区成员通过贡献代码、评论、讨论和其他形式的参与，感到参与社区是安全的
- 一定程度上相信存在相关政策来减少骚扰和不容忍
- 有一种机制可以有效地解决可能发生的问题
- 贡献者对这个项目没有严重的隐私泄露的担忧
- 在项目领导层和社区之间有很高的透明度
- 代码任何潜在的滥用途径已被项目领导作为优先考虑的问题

目标

该指标的目的是:

- 使项目维护人员能够了解围绕项目政策的心理安全水平，如行为准则，和行为准则的执行
- 允许社区成员就心理安全问题发表意见

- 确定社区内改善心理安全的潜在领域

实现

数据收集策略

下面是一个例子，通过一系列关于社区成员幸福感的有用问题，评估开源项目贡献者和维护者的心理安全水平。

候选问题包括:

1. 在开源项目的背景下，您是否观察到以下情况？答案: 是/否。如果选是 [选择适用的所有情况] :

- (1) 对贡献或问题缺乏答复
- (2) 未经解释拒绝接受贡献
- (3) 对贡献或问题不屑一顾的回答
- (4) 不完整或者难以理解的文件
- (5) 贡献者之间的冲突或人际关系紧张
- (6) 使你感到不受欢迎的语言或其他内容 (例如亵渎、种族主义笑话、色情图像等)

1. 在开源项目中，你有没有**目睹**过以下针对其他人的行为？（不包括直接针对你）答案: 是/否。如果选是 [选择适用的所有情况] :

- (1) 敌意或粗鲁
- (2) 骂人
- (3) 以暴力相威胁
- (4) 模仿
- (5) 长期的骚扰
- (6) 跨平台骚扰
- (7) 纠缠
- (8) 不请自来的性挑逗或评论
- (9) 基于人口统计学特征的刻板印象
- (10) 恶意发布个人信息 (doxxing)
- (11) 其他 (请描述)

1. 在开源项目中，你有没有**经历**过以下针对其他人的行为？（不包括直接针对你）答案: 是/否。如果选是 [选择适用的所有情况] :

- (1) 敌意或粗鲁
- (2) 骂人
- (3) 以暴力相威胁
- (4) 模仿
- (5) 长期的骚扰

- (6) 跨平台骚扰
- (7) 纠缠
- (8) 不请自来的性挑逗或评论
- (9) 基于人口统计学特征的刻板印象
- (10) 恶意发布个人信息 (doxxing)
- (11) 其他 (请描述)

3a. 如果对上述问题的回答是肯定的，那么当回想起上一次你遭遇的骚扰时，你是如何回应的呢？选择所有适用的。

- (1) 要求用户停止骚扰行为
- (2) 寻求其他社区成员的支持
- (3) 屏蔽骚扰我的用户
- (4) 向项目维护者举报事件
- (5) 向服务供应商举报事件
- (6) 咨询法律顾问/律师
- (7) 联络执法部门
- (8) 其他 (请描述)
- (9) 我没有对这件事作出反应/我对此置之不理

3b. 在 1-5 分的范围内，你的响应有多有效? (使用李克特量表量化下列选项) :

- 1: 完全没有效果
- 2: 有点效果
- 3: 略有成效
- 4: 大部分是有效的
- 5: 完全有效

1. 由于经历或目睹骚扰，如果有的话，你是否做过以下任何事情?

- (1) 停止对项目的贡献
- (2) 开始以假名贡献
- (3) 在私有渠道工作、提问或合作的次数较多
- (4) 更改或删除用户名
- (5) 删除或更改在线公开的内容
- (6) 建议制定或修改行为守则
- (7) 与社区成员就有关问题进行公开讨论
- (8) 与社区成员就有关问题进行私下讨论
- (9) 在我的线下生活做出了一些改变 (例如停止参加聚会或者会议等))
- (10) 其他 (请描述)

- (11) 以上皆非

其他可能包含的调查问题:

- 你是否觉得与其他项目贡献者或领导者分享的任何私人细节都将严格保密?
- 你是否觉得项目领导者重视参与者的安全?
- 你是否觉得项目领导者重视沟通和互动的透明度?
- 你是否觉得项目领导者能够接受关于信任和安全问题的批评性反馈?
- 项目领导者是否承认代码中有可能被滥用的潜在领域?
- 项目领导是否将这些问题作为优先事项来解决?
- 社区是否反对解决这些问题?

过滤条件

- 人口分布
- 贡献者的角色 (代码、社区管理和布道、文档等)
- 在社区时间长短

参考资料

- 组织中的心理安全、信任与学习: 一个群体层面的视角
- 为新功能添加社区和安全检查 (GitHub Blog)
- 2017 年开源调查
- 什么是工作中的心理安全

贡献者

- Elizabeth Barron
- Matt Germonprez
- Kevin Lumbard
- Lauren Phipps
- Dawn Foster
- Matt Cantu
- Justin W. Flory
- Emily Brown
- Amy Marrich
- Trisha Rajaram
- Ruth Ikegah
- Emily Brown
- Sean Goggins

- Georg Link

Evolution WG

关注领域	目标
代码开发活动	了解开发代码所涉及活动的类型和频率。
代码开发效率	了解如何有效地参与围绕代码开发的活动。
代码开发过程的质量	了解用于改进/审查质量的流程（例如：测试、代码审查、为议题打标签、为发布打标签、响应时间、CII Badging）。
议题解决	确定社区在解决社区参与者确定的议题方面的有效性。
社区成长	确定项目社区的规模，以及它是在增长、缩小还是保持不变。

关注领域 - 代码开发活动

目标: 了解开发代码所涉及活动的类型和频率。

度量指标	问题
分支的生命周期	项目如何管理其版本控制分支的生命周期?
代码变更	在指定时间内源代码变更了多少?
代码变更行数	在一段时间内对源代码的所有更改中，所触及的行数之和（增加的行数加上移除的行数）是多少?

分支的生命周期

问题: 项目如何管理其版本控制分支的生命周期?

描述

该指标使得版本控制分支的生命周期可见。分支生命周期包括分支创建和删除的行为，以及版本控制分支的持久性。在编写代码时，开发团队可能会创建多个分支，专注于某些特定功能。随后，这些分支可能会合并到更持久的分支，例如代码仓的主分支。一些分支在代码仓的生命周期内持续存在，而其他分支在代码合并到更持久的分支后被删除。通过了解这些模式，我们可以学习分支的创建、删除和合并如何反映特定项目的开发实践。一个代码仓的典型分支生命周期和管理方法可用于帮助识别一个项目的代码仓管理风格。

目标

在其他项目特征的背景下，该指标的目标是提高代码仓分支创建和删除的数量和频率的可见性，以及版本控制分支的持久性。反过来，这将有助于潜在贡献者了解如何发起和持续参与项目。可以通过该指标解决的问题包括:

- 有多少分支被合并但没有被删除?
- 一个分支被合并后多久还没有被删除?
- 有多少分支存在的时间超过一定的天数、月数或年数?
- 项目或代码仓多久创建一次分支?
- 总的来说，分支通常存活多长时间?
- 在代码仓分支很少被删除的情况下，我们如何区分分支 “死亡”（即不再打算再次使用；删除）或分支 “休眠”（即长时间不活动，但可能会再次使用）?

实现

可以在 CONTRIBUTING.md 文档中看到有关项目分支生命周期管理的所述建议。这些文档可以使用语言分析在代码仓之间进行比较，并与从实际项目实践中获得的数据进行对比，以在代码仓内部和跨代码仓中得出见解。然而，在大多数情况下，我们在这个指标中关注的数据是定量结果。

聚合器:

- 创建的分支数。
- 删除的分支数。
- 合并的分支数。
- 放弃的分支数（有提交，但在删除之前从未合并）
- 分支总数。
- 分支的平均生命周期。
- 创建分支与删除分支的比率。

参数:

- 时间段. 从开始到结束日期的时间段。默认：永久。
- 需要统计的代码变更所处的时期段。

筛选条件

- 代码仓集合
- 关于分支持久性的默认分支名称与描述性名称

数据收集策略

具体描述： *Git*

Git 分支数据存在于版本控制生态系统中的多个不同级别，通常同时存在于本地开发人员的电脑上以及托管平台（如 GitHub 或 BitBucket）上。托管平台上的这个分支数据也可能与每个开发者的电脑的数据不同，另外，不同的开发者在他们的电脑上可能有不同的分支数据，即使是同一个代码仓。

在 Git 特定情况下，由于 Git 设计为分布式版本控制系统，Git 允许单个存储库有多个远程仓（例如，用户在 github.com/user/project 上的 fork 克隆仓和 github.com/chaoss/project 上的上游仓），因此引入了大量额外的复杂性。通常情况下，许多个人贡献者可能在本地的同一个分支中工作，并将更改推送到远程仓中。因此，本地副本有时会与远程副本不同，远程副本可能在内部托管或在 GitHub、GitLab 和 BitBucket 等平台上托管。因为它们可能要么由不同的人管理（可能具有不同的分支风格），要么都被同一个人使用，以隔离区分不同的工作。

Git 分支的数据可以直接从 Git 日志中获取，也可以通过 Git 平台的 API 获取。

为分支数据选择一个 “规范” 版本的代码仓有用吗？这为比较具有不同版本的代码仓提供了一个有意义的基础。也可以使用

`git branch -a` 来列出所有存在的分支。

参考资料

采用 Git 分支策略: https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops

选择正确的 Git 分支策略: https://www.creativebloq.com/web-design/choose-right-git-branching-strategy-121518344

分支策略对软件质量的影响 Emad Shihab, Christian Bird, 和 Thomas Zimmermann https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/shihab-esem-2012.pdf

OpenStack: https://docs.openstack.org/project-team-guide/other-branches.html

代码变更

问题：在指定时间内源代码变更了多少？

描述

这些是指定时间内源代码的变更。对于 “变更”，我们会考虑开发者认为的代码的原子变更。换句话说，变更是对源代码的一些更改，通常作为一个整体被接受和合并，根据需要也会作为一个整体被还原。例如，对于 git，每个 “变更” 对应一个 “提交”，或者更准确地说，“代码变更” 对应于触及被视为源代码的文件的提交部分。

目标

- 编码活跃度。代码变更是项目活动度的一个维度。统计一个项目对应的代码仓库集合中的代码变更，可以让您了解该项目中的整体编码活动。当然，该指标不是唯一用于跟踪编码活跃度的指标。

实现

聚合器：

- 计数。一段时间内变更的总数。

参数：

- 时间段。开始日期和完成日期。默认：永久。需要统计的变更所处的时期段。
- 源代码标准。算法。默认：所有文件均为源代码。如果专注于源代码，需要一个标准来确定一个文件是否为源代码的一部分。

筛选条件

- 按参与者（作者、提交者）。需要将参与者合并（合并同一作者的对应 ID）。
- 按参与者群组（雇主、性别...）。需要将参与者分组，并且可能需要将参与者合并。
- 按标签（用于提交的消息）。需要提交消息的结构。在开源项目中，此标记可用于和每个贡献者交流，例如，提交是错误修复或功能改进的情况。

可视化效果

- 一段时间内每个月的计数
- 一段时间内每个群组的计数

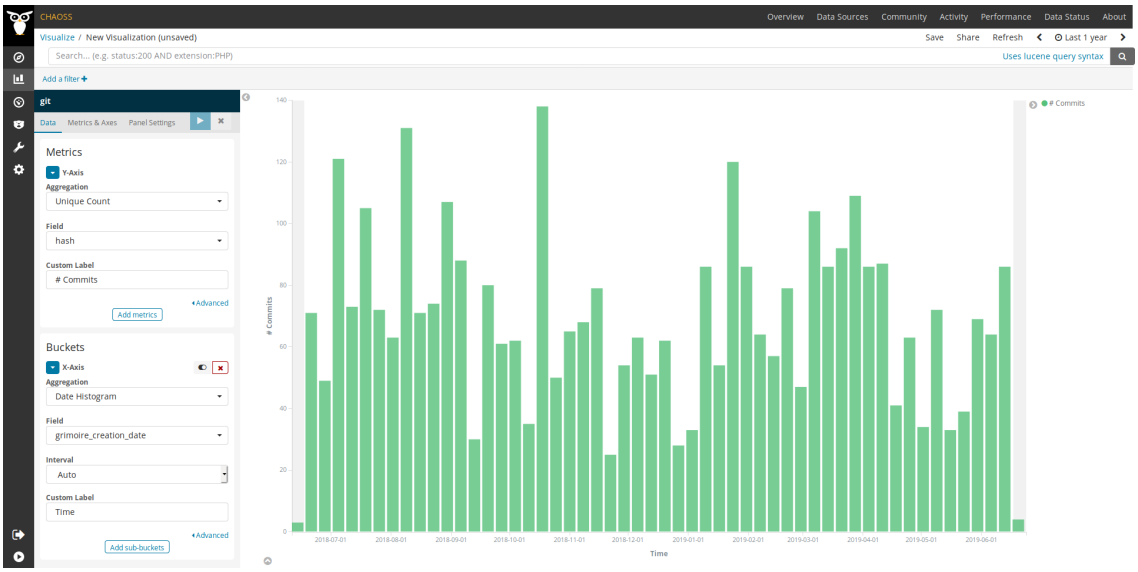
可以用条形图表示，X 轴为时间。每个条形代表一定时间（如一个月）内的代码变更。

提供指标的工具

- GrimoireLab 支持此指标，开箱即用。
  - 查看 Bitergia Analytics 的 CHAOSS 实例示例。
  - 从 GrimoireLab Sigils 面板集合下载并导入包含此指标可视化效果示例的现成仪表板。
  - 按照说明向任意 GrimoreLab Kibiter 仪表板添加一个示例可视化效果：
    - 新建一个 `Vertical Bar` 图

- 选择 `git` 索引
- Y 轴：`Unique Count` 聚合，`hash` 字段，`# Commits` 自定义标签
- X 轴：`Date Histogram` 聚合，`grimoire_creation_date` 字段，`Auto` 间隔，`Time` 自定义标签

◦ 屏幕截图示例：



- Augur 将此指标作为代码变更和代码变更行提供。两个指标均以 `repo` 和 `repo_group` 形式提供，更多信息见 Augur 文档。
- Gitdm

数据收集策略

具体描述：Git

请参阅 git 的参考实现

强制性参数（对于 Git）：

- 日期类型。作者日期或提交者日期。默认：作者日期。
  对于每个 git 提交，都会保留两个日期：撰写提交的日期和提交到仓库的日期。为了确定时间段，必须选择其中一项。
- 包括合并提交。布尔值。默认：True。
  合并提交是指合并一个分支的提交，在某些情况下不被认为反映编码活动。
- 包括空提交。布尔值。默认：True。
  空提交是未触及文件的提交，在某些情况下不被认为反映编码活动。

参考资料

- https://www.odoo.com/documentation/13.0/reference/guidelines.html#tag-and-module-name

代码变更行数

问题：在一段时间内对源代码的所有更改中，所触及的行数之和（增加的行数加上移除的行数）是多少？

描述

在引入对源代码的更改时，开发者会触及（编辑、添加、移除）源代码文件的多行修改。此指标考虑的是一段时间内对源代码更改所触及的行数的总和。这意味着，如果某个文件中的某一行在三次不同的变更中被触及，这将算作三行。由于在大多数源代码管理系统中，难以或无法准确判断一行是被移除后再被添加的还是仅被编辑的，所以我们将编辑一行视为移除后再添加新内容。（移除和添加）每一种都会被视为“触及”。因此，如果某个文件中的某一行被编辑了三次，则算作六次不同的变更（三次移除，三次添加）。

为此，我们认为源代码的变更如代码变更所定义。代码行将是源代码文件的任何一行，包括注释和空行。

目标

- 编码活跃度：虽然代码变更可以作为项目编码活跃度的一个维度，但并非所有变更都相同。关注所有变更中触及的行数的总和，可以补充说明变更规模，通常即为编码活跃度的规模。

实现

聚合器：

- 计数。一段时间内变更（触及）行的总数。

参数：

- ** 时间段：** 开始日期和完成日期。默认：永久。
  需要统计的变更所处的时期段。
- ** 源代码标准；算法默认：** 所有文件均为源代码。
  如果我们专注于源代码，则需要一个标准来确定一个文件是否为源代码的一部分。
- **源代码变更类型：**
  - 添加的行
  - 移除的行
  - 空格

筛选条件

- 按参与者（作者、提交者）。需要将参与者合并（合并同一作者的对应 ID）。
- 按参与者群组（雇主、性别...）。需要将参与者分组，并且可能需要将参与者合并。
- 按标签（用于提交的消息）。需要提交消息的结构。在开源项目中，此标记可用于和每个贡献者交流，例如，提交是错误修复或功能改进的情况。

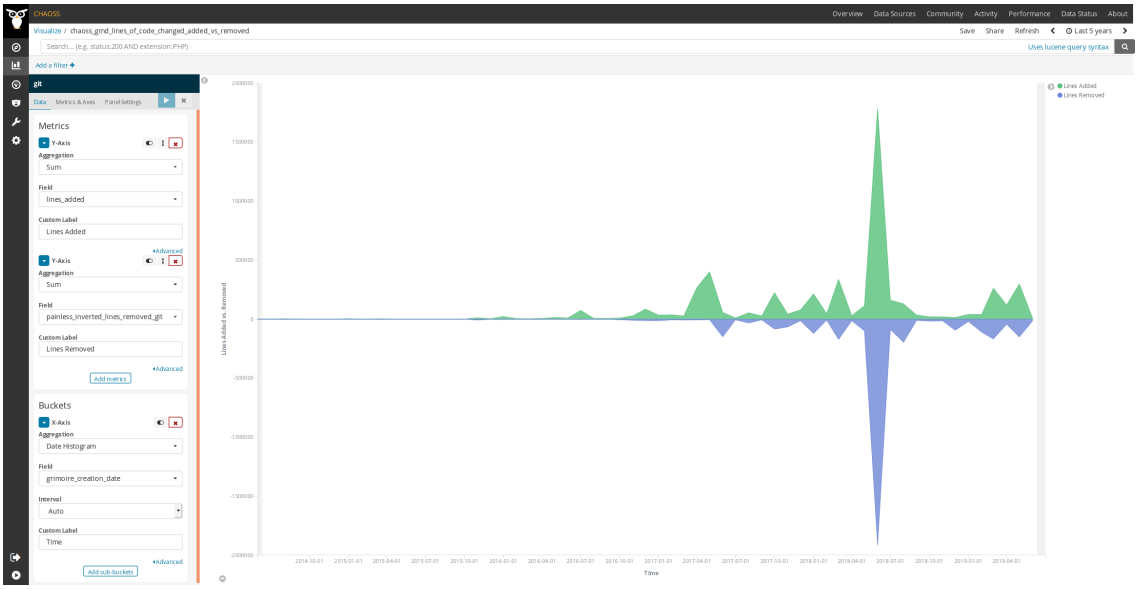
可视化效果

- 一段时间内每个月的计数
- 一段时间内每个群组的计数

可以用条形图表示，X 轴为时间。每个条形代表一定时间（如一个月）内的代码变更。

提供度量的工具

- GrimoireLab 支持此指标，开箱即用。
  - 查看 Bitergia Analytics 的 CHAOSS 实例示例。
  - 从 GrimoireLab Sigils 面板集合下载并导入包含此指标可视化效果示例的现成仪表板。
  - 按照说明向任意 GrimoreLab Kibiter 仪表板添加一个示例可视化效果：
    - 新建一个 `Area` 图
    - 选择 `git` 索引
    - Y 轴 1：`Sum` 聚合，`lines_added` 字段，`Lines Added` 自定义标签
    - Y 轴 2：`Sum` 聚合，`painless_inverted_lines_removed_git` 字段，`Lines Removed` 自定义标签
    - X 轴：`Date Histogram` 聚合，`grimoire_creation_date` 字段，`Auto` 间隔，`Time` 自定义标签
  - 屏幕截图示例：



数据收集策略

具体描述：Git

对于 git，“代码变更” 和 “变更日期” 的定义如代码变更所详述。变更的日期可以定义为作者日期或相应 git 提交的提交者日期（基于这两者情况发生在或不在一段时间内的考虑）。

由于 git 以差异补丁（添加和移除的行列表）的形式提供变更，所以差异中提及的每一个添加的行或移除的行都将被视为变更（触及）的行。如果行被移除后又被添加，则视为两次 “对行的变更”。

强制性参数：

- 日期类型。作者日期或提交者日期。默认：作者日期。
  对于每个 git 提交，都会保留两个日期：撰写提交的日期和提交到仓库的日期。为了确定时间段，必须选择其中一项。
- 包括合并提交。布尔值。默认：True。
  合并提交是指合并一个分支的提交，在某些情况下不被认为反映编码活动

参考资料

- https://www.odoo.com/documentation/13.0/reference/guidelines.html#tag-and-module-name

关注领域 - 代码开发效率

目标: 了解如何有效地参与围绕代码开发的活动。

度量指标	问题
接受的变更请求	在一次代码变更中存在多少接受的变更请求?
拒绝的变更请求	一段时间内，有哪些对代码变更的请求最终被拒绝变更?
变更请求时长	从代码变更请求开始到接受或关闭需要多长时间?
变更请求接受率	接受的变更请求与未合并而关闭的变更请求的比率是多少?

接受的变更请求

问题：在一次代码变更中存在多少接受的变更请求?

描述

变更请求如 变更请求中定义。接受的变更请求是指相应变更最终合并到项目代码库的变更请求。接受的变更请求可以链接到源代码的一个或多个变更，对应于提出后最终合并的变更。

例如，在 GitHub 中，当一个 pull request 被接受时，所有包含在其中的提交都会被合并（也许被 squashed，也许被 rebased）到相应的 git 仓库。GitLab 的 merge request 同样适用。对于 Gerrit，一次 code review 通常对应一次提交。

目标

- 编码活跃度。接受的变更请求是项目活动度的一个维度。统计一个项目对应的仓库集中接受的变更请求，可以让你了解该项目中导致实际变更的整体编码活动。当然，该指标不是唯一用于跟踪编码活跃度的指标。

实现

聚合器：

- 计数。一段时间内接受的变更请求的总数。
- 比率。一段时间内接受的变更请求占变更请求总数的比率。

参数：

- 时间段。考虑接受的变更请求的开始日期和结束日期。默认：永久。
- 源代码标准。算法。默认：所有文件均为源代码。
  如果我们专注于源代码，则需要一个标准来确定一个文件是否为源代码的一部分。

筛选条件

- 按参与者类型（提交者、审阅者、合并者）。需要合并同一参与者对应的身份。
- 按参与者群组（雇主、性别...涉及每个参与者）。需要参与者分组，并且可能需要参与者合并。

可视化效果

- 一段时间内每个周期的计数
- 一段时间内每个周期的比率

可以通过应用上述定义的筛选条件，按参与者类型或参与者群组进行分组。可以用条形图表示，X 轴为时间。每个条形代表一定时间（如一个月）内更改代码的接受的 reivew 。

提供度量的工具

- Grimoirelab 为 GitHub 拉取请求开箱即用地提供该指标，还为 GitLab 合并请求和 Gerrit 变更集提供构建类似可视化效果的数据。
  - 查看 Bitergia Analytics 的 CHAOSS 实例示例。

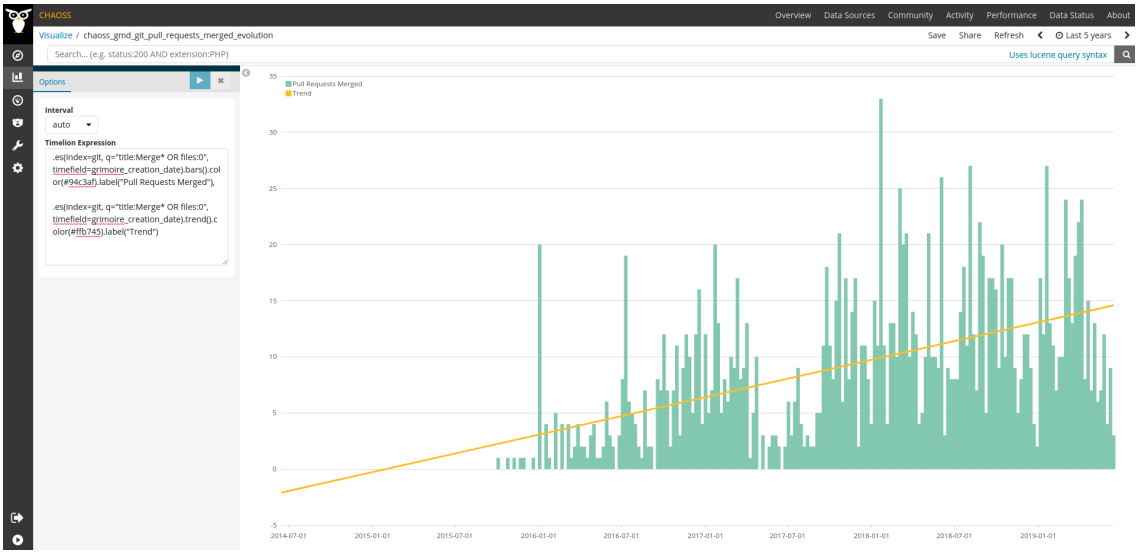
- 基于 GitHub 拉取请求数据，从 GrimoireLab Sigils 面板集合下载并导入包含此指标可视化效果示例的现成仪表板。
- 按照说明向任意 GrimoreLab Kibiter 仪表板添加一个 GitHub 拉取请求的示例可视化效果：
  - 新建一个 `Timelion` 可视化效果。
  - 选择 `Auto` 作为间隔。
  - 粘贴以下 Timelion 表达式：

```
.es(index=git, q="title:Merge* OR files:0", timefield=grimoire_creation_date).bars().c
```

  - 表达式的所有步骤：
    - `.es()`：用于定义 ElasticSearch 查询。
      - `index=git`：使用 git 索引。
      - `q="title:Merge* OR files:0"`：在合并中启发式筛选。
      - `timefield=grimoire_creation_date`：时间将基于提交的创建日期（因为我们的查询是寻找合并提交，所以应为合并有效完成的日期）。
    - `.bars()`：绘制条形而非线条。
    - `.color()` 和 `.label()`：一些格式设置选项。
  - 如果您还想要获得趋势，请改用（即重复两次相同的表达式，第二次调用 `trend()`）：

```
.es(index=git, q="title:Merge* OR files:0", timefield=grimoire_creation_date).bars().c
.es(index=git, q="title:Merge* OR files:0", timefield=grimoire_creation_date).trend().
```

  - 如上文所述 GitHub 情况，有时难以识别合并。您可能已经注意到，在此示例中，我们的表达式基于 GrimoireLab 的 Git 索引。此外，它还可以应用于使用 Git 仓库的任意相似环境，而不仅仅是 GitHub。
- 屏幕截图示例：



数据收集策略

具体描述：GitHub

在 GitHub 的情况下，接受的变更请求定义为 “变更包含在 git 仓库中的拉取请求”，前提是它提出对源代码文件的变更。

然而，接受变更请求的方式不止一种，并不是所有方式都能让人容易地识别出变更请求是否被接受。拉取请求被接受并合并是最简单的情况（或变基，或压缩并合并）。在这种情况下，可以很容易地确定拉取请求是否被接受，并通过对 GitHub API 的查询找到相应提交。

但也可以关闭变更请求，并在 git 仓库中手动合并提交。在这种情况下，提交仍然可以在 git 仓库中找到，因为其哈希值与拉取请求在 GitHub API 中的相同。

在更困难的情况下，也可以关闭变更请求，手动将提交变基，或压缩后合并。对于这些情况，哈希值是不同的，只能通过日期和作者的近似匹配和/或比较差异来跟踪 git 仓库中的提交。

从确定是否被接受的角度来看，问题在于如果其被手动包含在 git 仓库中，那么确定拉取请求被接受的唯一方法就是在 git 仓库中找到相应的提交。

在某些情况下，项目具有在关闭拉取请求时提及提交的政策（比如 “接受提交 xxx 和 yyyy 而关闭”），这可能有助于跟踪 git 仓库中的提交。

强制性参数（对于 GitHub）：

- 启发式检测未通过 web 接口接受的已接受拉取请求。默认：无。

具体描述：GitLab

在 GitLab 的情况下，接受的变更请求定义为 “变更包含在 git 仓库中的合并请求”，前提是它提出对源代码文件的变更。

强制性参数（对于 GitLab）：

- 启发式检测未通过 web 接口接受的已接受拉取请求。默认：无。

具体描述：Gerrit

在 Gerrit 的情况下，接受的变更请求定义为 “变更包含在 git 仓库中的变更集”，前提是其提出对源代码文件的变更。

强制性参数（对于 Gerrit）：无。

参考资料

拒绝的变更请求

问题：一段时间内，有哪些对代码变更的请求最终被拒绝变更？

描述

变更请求如变更请求中定义。拒绝的变更请求是指没有合并到项目代码库而最终关闭的变更请求。

例如，在 GitHub 中，当一个 pull request 在没有合并的情况下被关闭，而其中引用的提交在 git 仓库中找不到时，其可以被认为遭到拒绝（但请参阅以下详细讨论）。GitLab 的 merge request 同样适用。对于 Gerrit，代码变更请求可以正式 “放弃”，这也是该系统中检测拒绝的变更请求的方式。

目标

- 编码活跃度。拒绝的代码变更请求是项目活动度的一个维度。统计一个项目对应的仓库集合中被拒绝的代码变更请求，可以让您了解该项目中未导致实际变更的整体编码活动。当然，该指标不是唯一用于跟踪编码活跃度的指标。

实现

聚合器：

- 计数。一段时间内拒绝的变更请求的总数。
- 比率。一段时间内拒绝的变更请求占变更请求总数的比率。

参数：

- 时间段。考虑拒绝的变更请求的开始日期和结束日期。默认：永久。
- 源代码标准。算法。默认：所有文件均为源代码。
  如果我们专注于源代码，则需要一个标准来确定一个文件是否为源代码的一部分。

筛选条件

- 按参与者（提交者、审阅者、合并者）。需要合并同一参与者对应的身份。
- 按参与者群组（雇主、性别...涉及每个参与者）。需要参与者分组，并且可能需要参与者合并。

可视化效果

- 一段时间内每个周期的计数
- 一段时间内每个周期的比率

这些数据可以通过应用筛选条件进行分组（按参与者类型或按参与者群组），并且可以表示为条形图，X 轴为时间。每个条形代表一定时间（如一个月）内被拒绝的审查。

数据收集策略

具体描述：GitHub

在 GitHub 的情况下，拒绝的变更请求定义为 “关闭的拉取请求，其变更未包含在 git 仓库中”，前提是它提出对源代码文件的变更。

请参阅接受的变更请求中关于 GitHub 具体描述的讨论，因其在此处也适用。

强制性参数（对于 GitHub）：

- 用于检测拒绝的拉取请求的启发式方法，区分那些拉取请求被关闭但变更被手动包含在 git 仓库中的情况。默认：无。

具体描述：GitLab

在 GitLab 的情况下，拒绝的变更请求定义为 “关闭的合并请求，其变更未包含在 git 仓库中”，前提是它提出对源代码文件的变更。

强制性参数（对于 GitLab）：

- 用于检测拒绝的合并请求的启发式方法，区分那些合并请求被关闭但变更被手动包含在 git 仓库中的情况。默认：无。

具体描述：Gerrit

在 Gerrit 的情况下，拒绝的变更请求定义为 “放弃的变更集”，前提是其提出对源代码文件的变更。

强制性参数（对于 Gerrit）：无。

参考资料

变更请求时长

问题：从代码变更请求开始到接受或关闭需要多长时间?

描述

变更请求如变更请求中定义。接受的变更请求的定义位于接受的变更请求中。

变更请求时长是指从代码变更请求开始到结束（被接受并合并到代码库中）这段时间的长度。这仅适用于接受的变更请求。

例如在 GitLab 中，当开发者上传提案的代码变更时，合并请求开始，同时打开合并请求。当提交最终被接受并合并到代码库中时，即结束，同时关闭合并请求。

如果在代码合并后有评论或其他事件，不考虑将这些评论或其他事件用于衡量代码变更请求的时长。

目标

- 接受变更请求的时长。接受的变更请求的代码审查时长指标能够表示项目在接受代码贡献前需要经过多长时间。当然，该指标不是唯一用于跟踪编码活跃度的指标。

实现

聚合器：

- 中位数。在考虑的时间段内，所有接受的变更请求的变更请求时长的中位数 (50%)。

参数：

- 时间段。开始日期和完成日期。默认：永久。
  考虑接受的变更请求的时间段。如果接受的变更请求的创建事件在该时间段内，则接受的变更请求在该时间段内。
- 源代码标准。算法。默认：所有文件均为源代码。
  如果我们专注于源代码，则需要一个标准来确定一个文件是否为源代码的一部分。

筛选条件

- 按参与者（提交者、审阅者、合并者）。需要参与者合并（合并同一作者的对应 ID）。
- 按参与者群组（雇主、性别...涉及每个参与者）。需要参与者分组，并且可能需要参与者合并。

可视化效果

- 一段时间内的每月中位数
- 一段时间内的每组中位数

可以用条形图表示，X 轴为时间。每个条形代表一定时间（如一个月）内更改代码的接受的变更请求。

- 一定时期的时长分布

这些可以表示为通常的统计分布曲线或条形图，在 X 轴用桶表示时长，在 Y 轴为变更请求的数量。

数据收集策略

具体描述：GitHub

对于 GitHub，考虑在代码库中接受并合并的拉取请求的时长。对于单独的拉取请求，时长从打开时开始，直到其包含的提交被合并到代码库时结束。

强制性参数（对于 GitHub）：无。

具体描述：GitLab

对于 GitLab，考虑在代码库中接受并合并的合并请求的时长。对于单独的合并请求，时长从打开时开始，直到其包含的提交被合并到代码库时结束。

强制性参数（对于 GitLab）：无。

具体描述：Gerrit

对于 Gerrit，考虑在代码库中接受并合并的代码变更请求的时长。对于单独的代码变更请求，时长从打开时开始，直到其包含的提交被合并到代码库时结束。

强制性参数（对于 Gerrit）：无。

参考资料

变更请求接受率

问题: 接受的变更请求与未合并而关闭的变更请求的比率是多少?

描述

每个变更请求都可以处于以下三种状态之一：打开、合并（接受）和关闭而不合并（拒绝）。此指标衡量合并（接受）的变更请求与关闭但未合并的变更请求的比率。

目标

合并的变更请求与未合并而关闭的变更请求的比率提供了对多个代码仓特征的洞察，包括对外部贡献的开放性、贡献者社区的发展、代码审查过程的效率，以及随着时间的推移衡量项目发展的轨迹。应该基于每个代码仓或项目来诠释比率的差异。

实现

参数

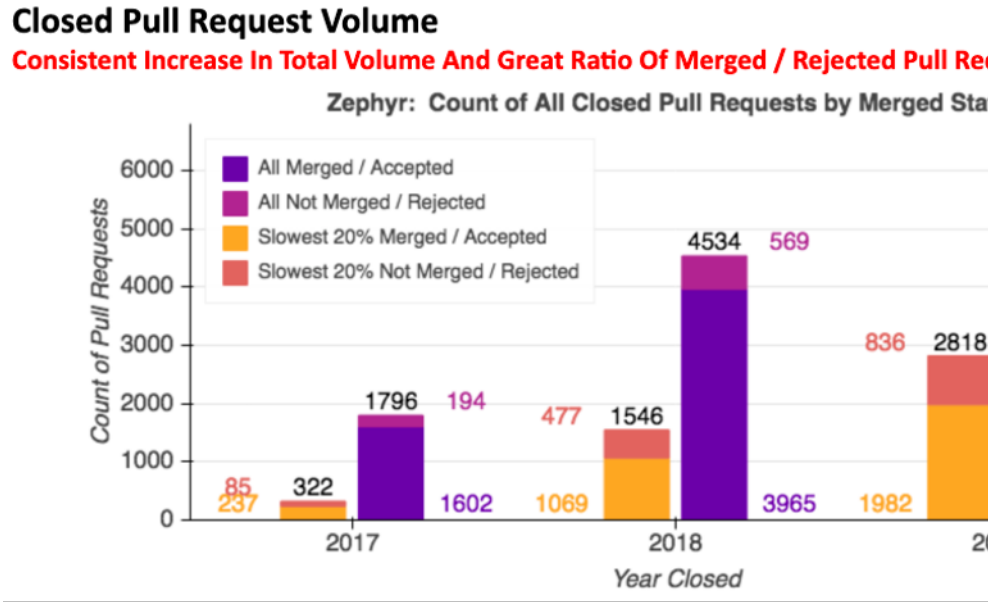
- 时间段粒度 (每周, 每月, 每年)。
- 一段时间内的比率变化。
- 显示贡献者数量
- 变更请求的来源：branch 还是 fork？来自代码仓 fork 的变更请求更常见于外部贡献者，而由 branch 发起的变更请求来自具有代码仓提交权限的人

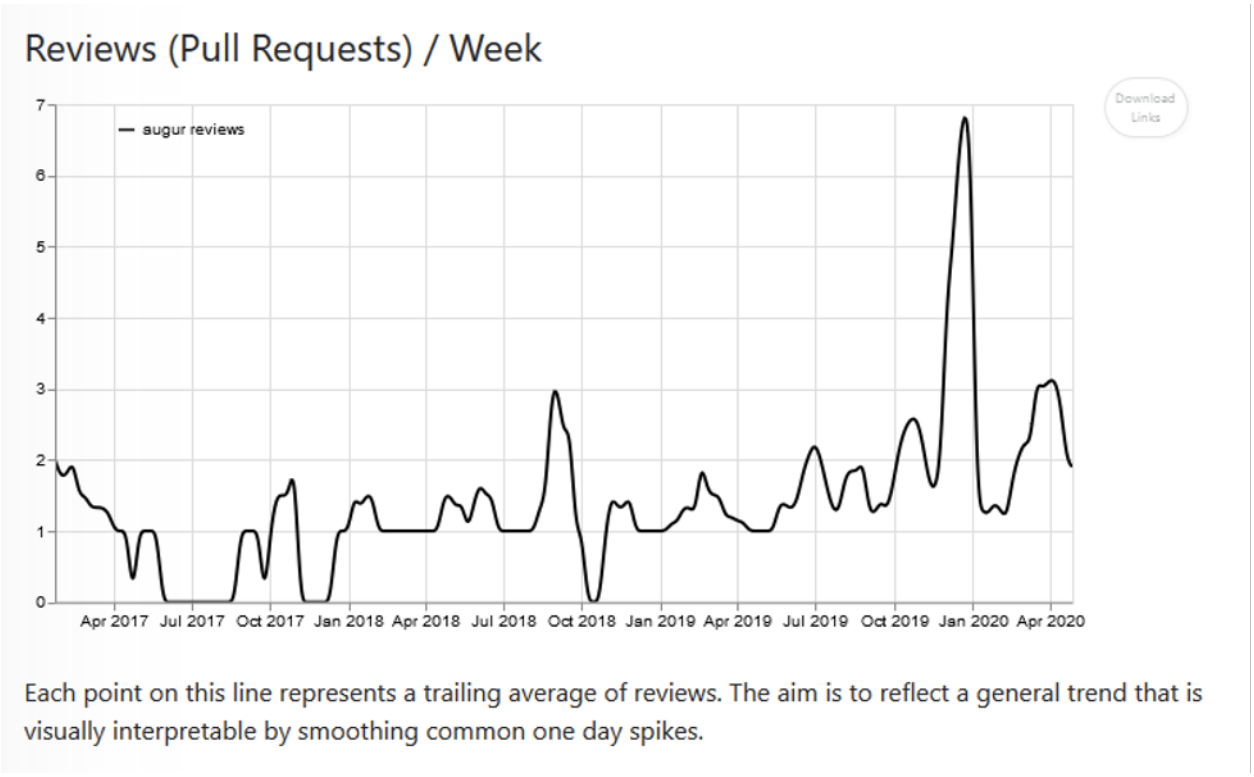
* **聚合器**

- 合并的变更请求总数（已接受）
- 未合并而关闭的变更请求总数
- 处于打开状态的变更请求总数

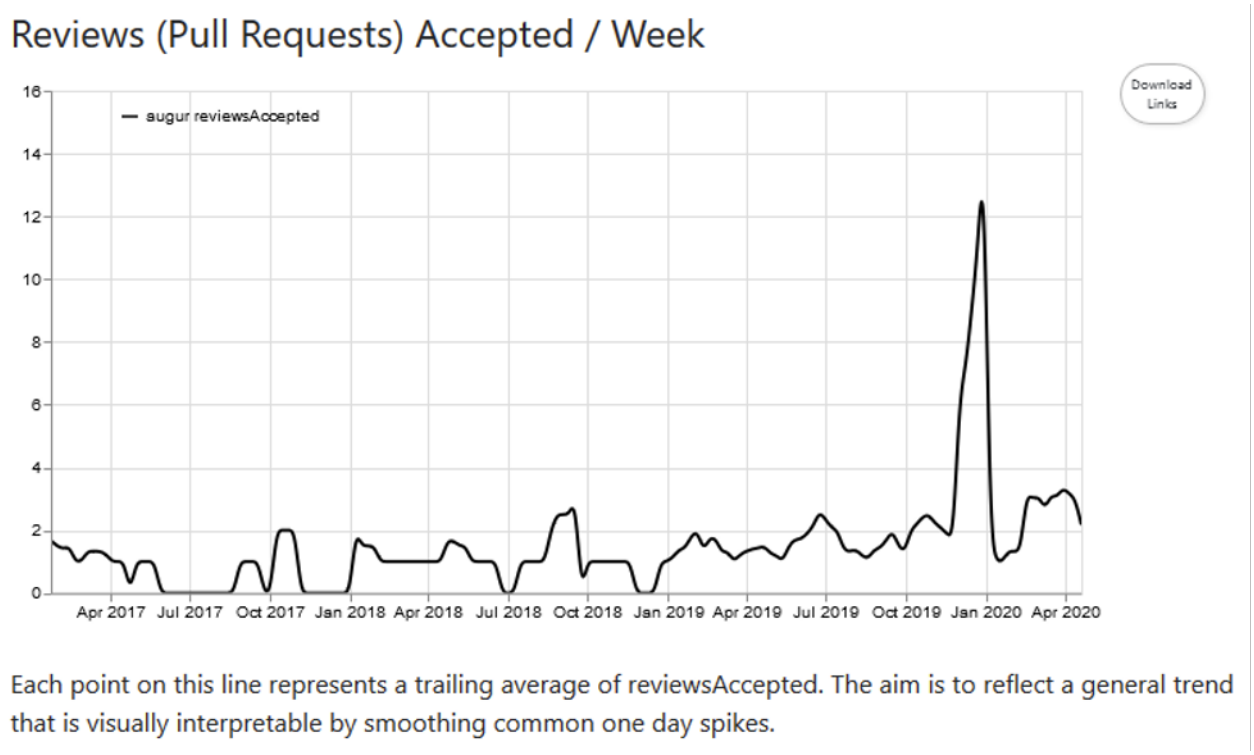
可视化效果

CHAOSS 工具为此指标提供了许多可视化图形。第一个可视化显示了每年接受和拒绝的变更请求,从中可以得出比率。图形 1:





图形 2:



图形 3:

提供度量的工具

- https://github.com/chaoss/augur
- https://github.com/chaoss/augur-community-reports

数据收集策略

接受的变更请求 Change Requests Accepted 指标, 以及拒绝的变更请求 Change Requests Declined 指标。

参考资料

Augur Zephyr 关于拉取请求的报告: https://docs.google.com/presentation/d/11b48Zm5Fwsmd1OlHg4bse5ibaVJUWkU
edit#slide=id.g7ec7768776_1_56

关注领域 - 代码开发过程的质量

目标: 了解用于改进/审查质量的流程（例如：测试、代码审查、为议题打标签、为发布打标签、响应时间、CII Badging）。

度量指标	问题
变更请求	在一段时间内发生了哪些新的变更源代码的请求?

变更请求

问题：在一段时间内发生了哪些新的变更源代码的请求？

描述

当项目采用代码变更请求流程时，变更并不直接提交至代码库，而是先以“源代码变更提案”的形式进行讨论。每一次提交都应由其他开发者审查，这些开发者可能会提出改进建议，让原始提议者发送新版本的提案，直到取得肯定的审查结果，接受代码，或者直到提案被拒绝。

例如，“变更请求”在 GitHub 中对应“拉取请求（Pull Request）”，在 GitLab 中对应“合并请求（Merge Request）”，在 Gerrit 中对应“代码审查（Code Review）”，或在某些语境对应“变更集（changesets）”。

目标

- 项目中变更请求的数量。变更请求是项目活动度的一个维度。统计一个项目对应的代码仓集合中的变更请求数量，可以让你了解该项目中变更的整体活跃度。当然，该指标不是唯一用于跟踪编码活跃度的指标。

实现

聚合器：

- 计数。一段时间内变更请求的总数。

参数：

- 时间段。开始日期和完成日期。默认：永久。
  需要统计的变更所处的时期段。
- 源代码标准。算法。默认：所有文件均为源代码。
  如果我们专注于源代码，则需要一个标准来确定一个文件是否为源代码的一部分。

筛选条件

- 按参与者（提交者、审阅者、合并者）。需要将参与者合并（合并同一作者的对应 ID）。
- 按参与者群组（雇主、性别...涉及的每个参与者）。需要将参与者分组，并且可能需要参与者合并。
- 状态（打开，关闭）

可视化效果

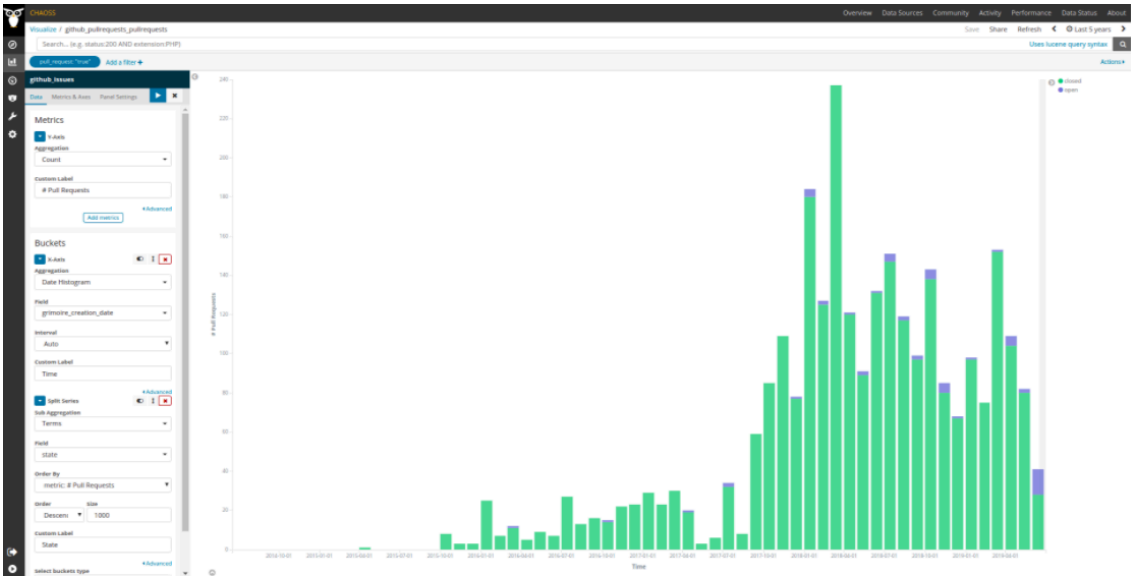
- 一段时间内每个月的计数
- 一段时间内每个群组的计数

可以用条形图表示，X 轴为时间。每个条形代表一定时间（如一个月）内更改代码的变更请求。

提供指标的工具

- Grimoirelab 为 GitHub 拉取请求、GitLab 合并请求和 Gerrit 变更集开箱即用地提供该指标。
  - 查看 Bitergia Analytics 的 CHAOSS 实例示例。

- 基于 GitHub 拉取请求数据，从 GrimoireLab Sigils 面板集合下载并导入包含此指标可视化效果示例的现成仪表板。
- 屏幕截图示例：



数据收集策略

具体描述：GitHub

对于 GitHub，变更请求定义为“拉取请求（Pull Request）”，前提是其提出对源代码文件的变更。

变更请求的日期可以定义为（基于这两者情况发生在或不在一段时间内的考虑）拉取请求的提交日期。

具体描述：GitLab

对于 GitLab，变更请求定义为“合并请求（Merge Request）”，前提是其提出对源代码文件的变更。

变更请求的日期可以定义为（基于这两者情况发生在或不在一段时间内的考虑）合并请求的提交日期。

具体描述：Gerrit

对于 Gerrit，定义为“代码审查（Code Review）”，某些上下文中为“变更集（changesets）”，前提是其提出对源代码文件的变更。

变更请求的日期可以定义为（基于这两者情况发生在或不在一段时间内的考虑）通过提交补丁集进行变更请求而开始审查的日期。

参考资料

关注领域 - 议题解决

目标: 确定社区在解决社区参与者确定的议题方面的有效性。

度量指标	问题
新议题	在一定时期内创建了多少个新议题?
活跃议题	在一定时期内有多少活跃议题?
关闭的议题	在一定时期内有多少关闭的议题?
议题时长	未解决议题有多长时间处于未解决状态?
议题响应时间	从议题创建到另一贡献者对议题做出响应经过了多长时间?
议题解决时长	解决议题花费了多长时间?

新议题

问题：在一定时期内创建了多少个新议题?

描述

项目在议题跟踪系统中讨论如何修复错误或增加新功能。每一个议题都由某人创建（提交），随后被很多人评论和注解。

根据考虑的议题系统，一个议题可能会经历多种状态（如 “已分类”、“工作中”、“已修复”、“不可修复”），或带有一个或多个标记，或被分配给一个或多个人。但在任何议题跟踪系统中，议题通常都是评论和状态变更的集合，也许还带有其他注解。在某些系统中，议题也可以与里程碑、分支、epic 或 story 相关。在某些情况下，其中一些也是议题本身。

通常至少可以确定两种 “高级” 状态：打开和关闭。“打开” 通常表示议题尚未解决，“关闭” 表示议题已经解决，不会再对其做进一步的工作。然而，如何确定一个议题是 “开放” 还是 “关闭”，在某种程度上取决于议题跟踪系统，以及特定项目如何使用该系统。在实际项目中，难以筛选与源代码直接相关的议题，因为议题跟踪系统可能被用于多种信息，涵盖修复错误和讨论新功能的实现、组织项目事件或询问如何使用项目结果。

在大多数议题跟踪器中，议题被关闭后还可重新打开。重新打开议题可以被视为打开一个新的议题（见下文参数）。

例如，“议题” 在 GitHub、GitLab 或 Jira 中对应 “issue”，在 Bugzilla 中对应 “bug reports”，在其他系统中对应 “issues” 或 “tickets”。

目标

项目中讨论的议题量。议题是项目活动度的一个维度。统计一个项目对应的仓库集合中讨论代码的议题数，可以了解该项目中讨论议题的整体活跃度。当然，该指标不是唯一用于跟踪编码活跃度的指标。

实现

聚合器：

- 计数。一段时间内新议题的总数。
- 比率。一段时间内新议题占议题总数的比率。

参数：

- 时间段。考虑议题的开始日期和结束日期。默认：永久。
- 源代码标准。算法。默认：所有议题均与源代码相关。
  如果我们专注于源代码，则需要一个标准来确定一个议题是否与源代码相关。
- 重新打开为新议题。布尔值。默认：False。判断重新打开的议题是否为新议题的标准。

筛选条件

- 按参与者（提交者、评论者、解决者）。需要合并同一作者对应的身份。
- 按参与者群组（雇主、性别...涉及每个参与者）。需要参与者分组，并且可能需要参与者合并。

可视化效果

- 一段时间内每个周期的计数
- 一段时间内每个周期的比率

这些可以通过应用上述定义的筛选条件进行分组。可以用条形图表示，X 轴为时间。每个条形代表一定时间（如一个月）内更改代码的提案。

数据收集策略

具体描述：GitHub

对于 GitHub，议题定义为 “议题”。

议题的日期可以定义为（为了在或不在一段时间内考虑）议题的创建（提交）日期。

具体描述：GitLab

对于 GitHub，议题定义为 “议题”。

议题的日期可以定义为（为了在或不在一段时间内考虑）议题的创建（提交）日期。

具体描述：Jira

对于 Jira，议题定义为 “议题”。

议题的日期可以定义为（为了在或不在一段时间内考虑）议题的创建（提交）日期。

具体描述：Bugzilla

对于 Bugzilla，议题定义为 “错误报告”，前提是其与源代码文件相关。

议题的日期可以定义为（为了在或不在一段时间内考虑）错误报告的创建（提交）日期。

参考资料

活跃议题

问题：在一定时期内有多少活跃议题?

描述

议题如新议题中定义。显示某些活动的议题是指在一定时期内具有一些注释，或状态发生一些变化（包括关闭议题）的议题。

例如，在 GitHub 议题中，注释、新标记或关闭议题的操作，都被认为是活动的标志。

目标

- 项目中的活跃议题量。活跃议题是项目活动度的一个维度。统计一个项目对应的代码仓集合中与代码相关的活跃议题，可以了解该项目中处理议题的整体活跃度。当然，该指标不是唯一用于跟踪编码活跃度的指标。

实现

聚合器：

- 计数。一段时间内活跃议题的总数。
- 比率。一段时间内活跃议题占议题总数的比率。

参数：

- 时间段。考虑议题的开始日期和结束日期。默认：永久。
- 源代码标准。算法。默认：所有议题均与源代码相关。
  如果我们专注于源代码，则需要一个标准来确定一个议题是否与源代码相关。

筛选条件

- 按参与者（提交者、评论者、解决者）。需要合并同一作者对应的身份。
- 按参与者群组（雇主、性别...涉及每个参与者）。需要参与者分组，并且可能需要参与者合并。

可视化效果

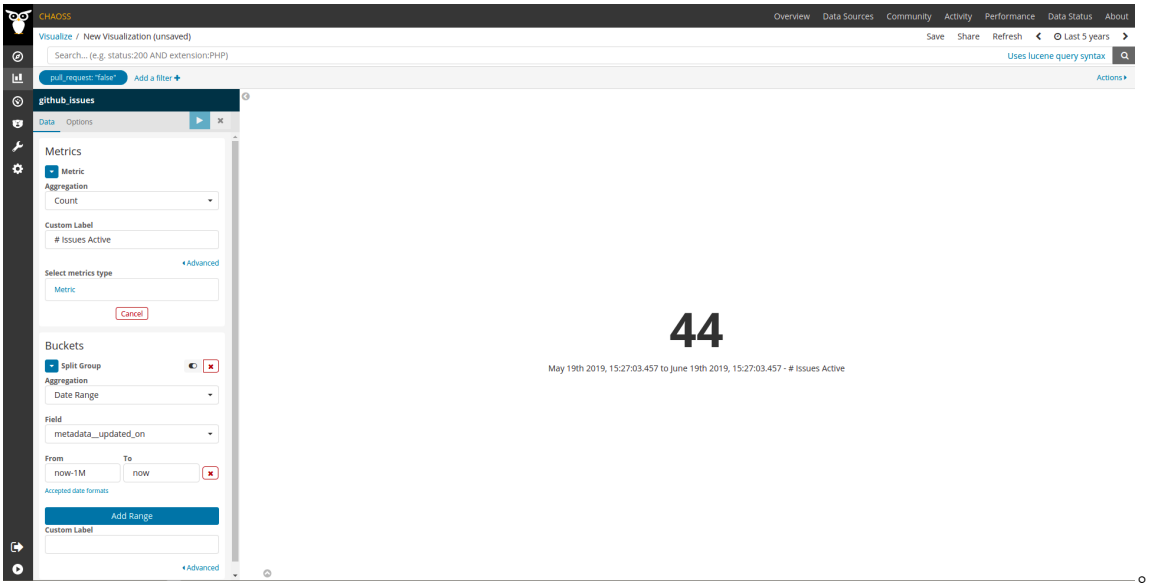
- 一段时间内每个周期的计数
- 一段时间内每个周期的比率

这些可以通过应用先前定义的筛选条件进行分组。可以用条形图表示，X 轴为时间。每个条形代表一定时间（如一个月）内更改代码的提案。

提供指标的工具

- GrimoireLab 为 GitHub 议题、GitLab 议题、Jira、Bugzilla 和 Redmine 提供了计算此页面相似指标的数据。就指标而言，**GrimoireLab 数据只有每个项的最后更新日期，这将计算指标的时间范围限制到当前日期为止**。
  - 根据源 API 的不同，对议题更新的定义可能有所不同。GrimoireLab 使用 `metadata__updated_on` 存储最新议题更新，请查看 Perceval 文档查找每种情况下正在使用的特定 API 字段，并了解其限制（如果有）。

- 当前，没有显示此操作的仪表板。不过，构建可视化效果仍很容易，可以显示上一次活动在给定日期和当前日期之间某个时刻的使用次数（我们将针对 GitHub 议题进行操作）。
- 按照说明向任意 GrimoreLab Kibiter 仪表板添加一个示例可视化效果：
  - 新建一个 `Metric` 可视化效果。
  - 选择 `github_issues` 索引。
  - 筛选条件：`pull_request` 为 `false` 。
  - 指标：`Count` 聚合，`# Issues Active` 自定义标签。
  - 桶：`Date Range` 聚合，`metadata__updated_on` 字段，`now-1M` 自从（或任何适合您需求的间隔），`now` 直到，将“自定义标签”留空以查看图例中的具体日期。
  - 查看右上角的时间选择器，确保将其设置为包括数据的整个故事，这样我们就不会根据其创建日期排除任何项目。
- 屏幕截图示例：



数据收集策略

具体描述：GitHub

对于 GitHub，活跃议题定义为“获得注释、标记变更、分配人员变更或被关闭的议题”。

具体描述：GitLab

对于 GitLab，活跃议题定义为“获得注释、标记变更、分配人员变更或被关闭的议题”。

具体描述：Jira

对于 Jira，活跃议题定义为“获得注释、状态变更、分配人员变更或被关闭的议题”。

具体描述：Bugzilla

对于 Bugzilla，活跃议题定义为“获得注释、状态变更、分配人员变更或被关闭的错误报告”。

参考资料

关闭的议题

问题：在一定时期内有多少关闭的议题?

描述

议题如新议题中定义。关闭的议题是指在一定时期内变更为关闭状态的议题。

在某些情况下或某些项目中，还有其他状态或标记可以被视为“关闭”。例如，在某些项目中，使用状态或标记 “fixed” 说明议题已关闭，即使它还需要一些操作才能正式关闭。

在大多数议题跟踪器中，议题被关闭后还可重新打开。重新打开议题可以被视为打开一个新的议题，或者使先前的关闭无效（见下文参数）。

例如，在 GitHub 议题或 GitLab 议题中，关闭的议题是指在某一时期关闭的议题。

目标

项目中处理的议题量。关闭的议题是项目活动度的一个维度。统计一个项目对应的代码仓集合中与代码相关的关闭的议题，可以了解该项目中完成议题处理的整体活跃度。当然，该指标不是唯一用于跟踪编码活跃度的指标。

实现

聚合器：

- 计数。一段时间内活跃议题的总数。
- 比率。一段时间内活跃议题占议题总数的比率。
- 反馈。对议题点赞或其他反馈类型的次数。

参数：

- 时间段。考虑议题的开始日期和结束日期。默认：永久。
- 源代码标准。算法。默认：所有议题均与源代码相关。
  如果我们专注于源代码，则需要一个标准来确定一个议题是否与源代码相关。通过更改默认值，可以将所有议题都包含在指标中。
- 重新打开为新议题。布尔值，定义是否将重新打开的议题视为新议题。如果为 false，则意味着重新打开之前的关闭操作应视为无效。注意：如果该参数为 false，则任何时期的关闭的议题数量都可能会在未来发生变化，如果其中一些议题被重新打开。
- 关闭标准。算法。默认：在相关时间内有关闭事件。

筛选条件

- 按参与者（提交者、评论者、解决者）。需要合并同一作者对应的身份。
- 按参与者群组（雇主、性别...涉及每个参与者）。需要参与者分组，并且可能需要参与者合并。

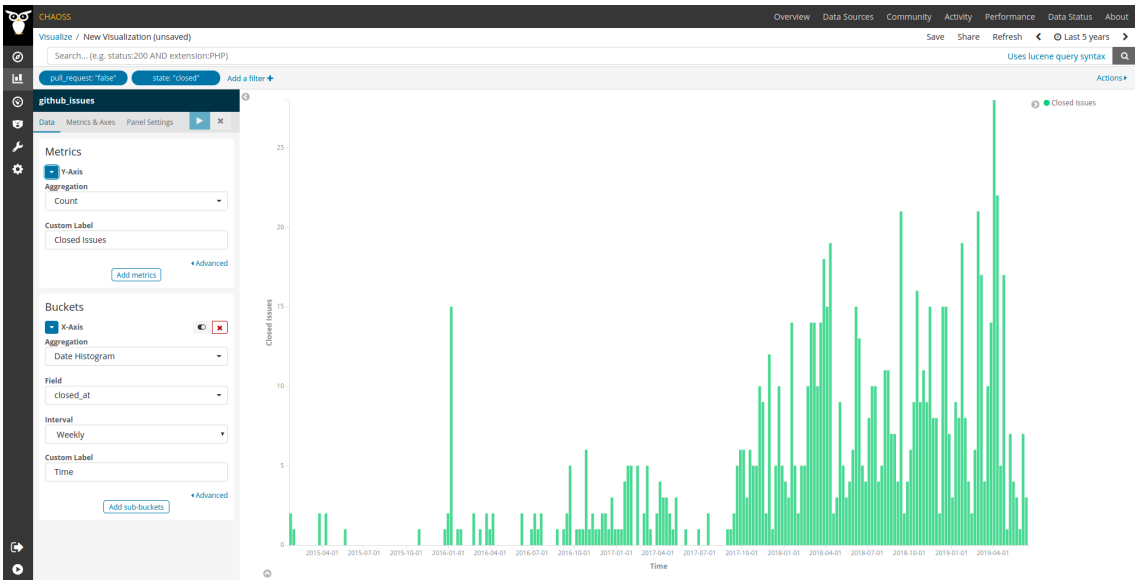
可视化效果

- 一段时间内每个周期的计数
- 一段时间内每个周期的比率

这些可以通过应用上述定义的筛选条件进行分组。可以用条形图表示，X 轴为时间。

提供指标的工具

- GrimoireLab 为 GitHub 议题、GitLab 议题、Jira、Bugzilla 和 Redmine 提供了计算指标的数据。当前仪表板根据创建日期显示信息，这意味着其显示了在某个时间段内创建的议题的当前状态（例如 GitHub 议题仪表板，您可以查看其运作中的状态）。不过，通过执行以下步骤，仍然可以轻松建立可视化效果基于关闭日期显示议题：
  - 按照说明向任意 GrimoreLab Kibiter 仪表板添加一个示例可视化效果：
    - 新建一个 `Vertical Bar` 图。
    - 选择 `github_issues` 索引。
    - 筛选条件：`pull_request` 为 `false` 。
    - 筛选条件：`state` 为 `closed` 。
    - 指标 Y 轴：`Count` 聚合，`# Closed Issues` 自定义标签。
    - 桶 X 轴：`Date Histogram` 聚合，`closed_at` 字段，`Weekly` 间隔（或任何适合您需求的间隔，取决于您希望在图表中可视化的整个时间范围），`Time` 自定义标签。
  - 屏幕截图示例：



数据收集策略

具体描述：GitHub

对于 GitHub，关闭的议题定义为 “关闭的议题”。

具体描述：GitLab

对于 GitLab，关闭的议题定义为 “关闭的议题”。

具体描述：Jira

对于 Jira，关闭的议题定义为 “变更为关闭状态的议题”。

具体描述：Bugzilla

对于 Bugzilla，关闭的议题定义为 “变更为关闭状态的错误报告”。

参考资料

议题时长

问题：未解决议题有多长时间处于未解决状态?

描述

这一指标表明在考虑的时间段内，议题有多长时间处于未解决状态。如果一个议题已解决，但在这段时间内又重新变为未解决，则将其视为自最初发布为未解决的日期以来一直为未解决状态。

目标

当议题时长不断增加时，找出项目中时间最长的未解决议题，深入了解为什么这些议题长时间都未能解决。此外，要了解维护者解决议题的水平和速度。

实现

对于所有未解决议题，获取议题发布为未解决的日期，计算距当前日期的天数。

聚合器：

- 平均数。所有未解决议题的平均时长。
- 中位数。所有未解决议题的中位时长。

参数：

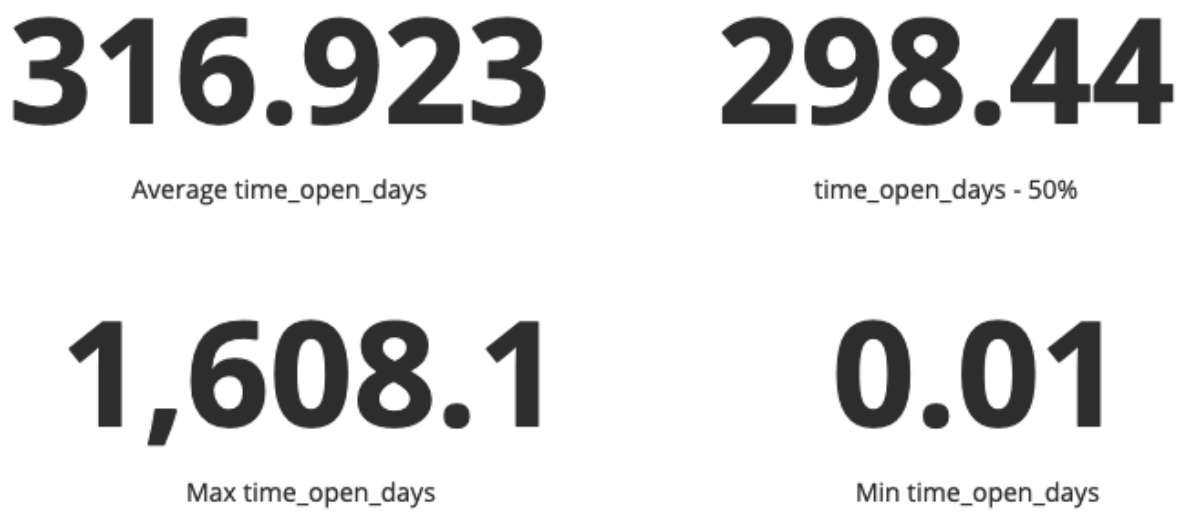
- 时间段。考虑未解决议题的开始日期和结束日期。默认：永久（即项目议题活动的整个可观察时段）。

筛选条件

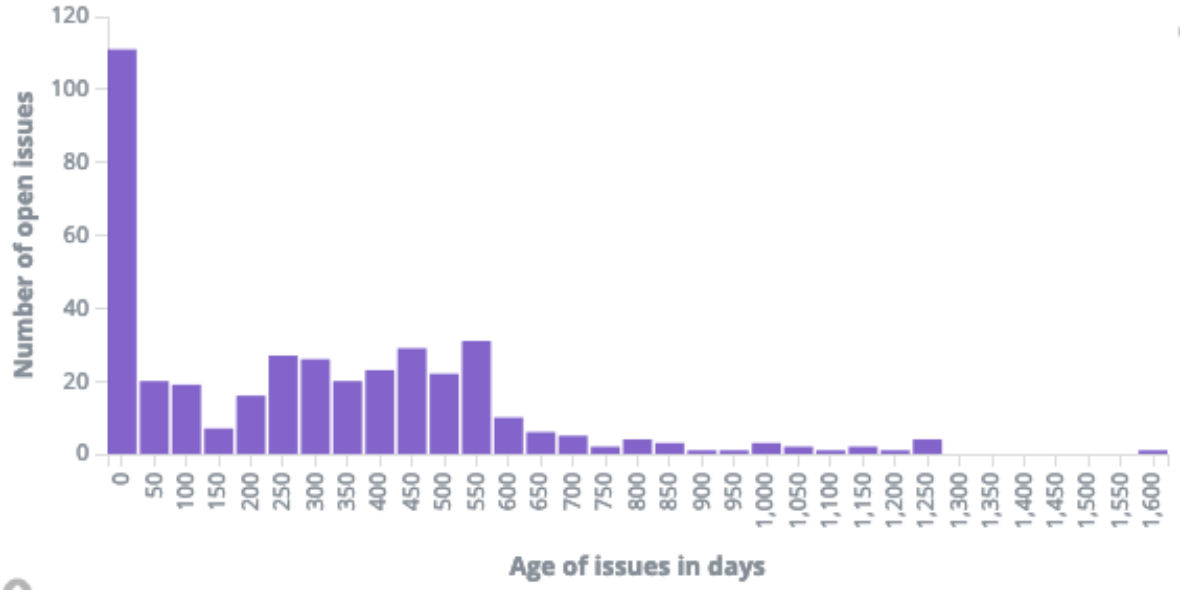
- 模块或工作组
- 议题标记/标签

可视化效果

1. 未关闭的 issue 汇总数据



2. 每天未关闭 issue 的数量



提供度量的工具

- GrimoireLab
- Augur

数据收集策略

关于收集已关闭 issue 数据的具体说明，请参阅“新 issue” 的对应部分。

参考资料

议题响应时间

问题：从议题创建到另一贡献者对议题做出响应经过了多长时间？

描述

这一指标表明从议题创建到其他贡献者响应之间的时间间隔。

这一指标是共同工作组中第一响应时长的特定情况。

目标

了解开源社区的响应能力。

实现

聚合器：

- 平均数。平均响应时间（天）。
- 中位数。中位响应时间（天）。

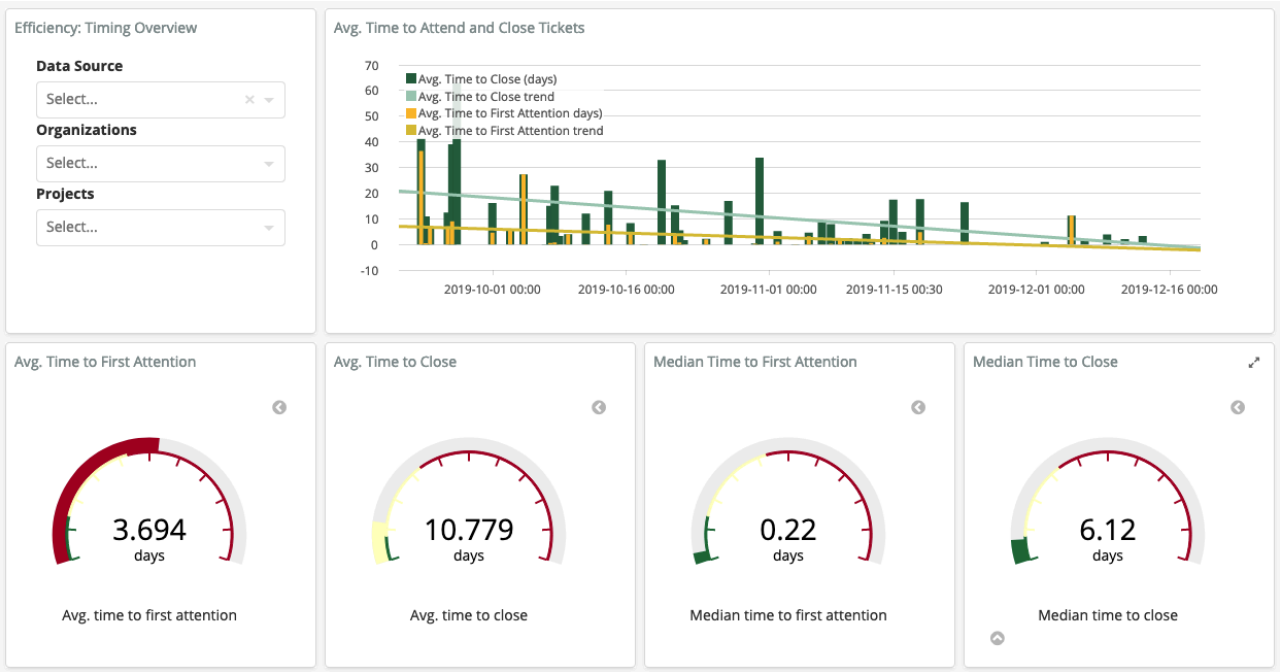
参数：

- 时间段。开始日期和完成日期。默认：永久。
- 计算响应的时期。

筛选条件

- 项目角色的响应（例如，第一维护者响应）
- 机器人与人类（例如，滤除自动化的 “欢迎第一贡献者消息”）
- 由项目中的角色创建（例如，对新贡献者的响应与长期贡献者的响应）
- 创建议题的日期
- 议题的状态（例如，仅查看当前未解决议题）

可视化效果



提供指标的工具

- GrimoireLab：任意票证系统的一般情况、GitHub 议题、GitLab 议题
- Augur

数据收集策略

查看新议题指标的 “议题” 定义。从第一响应时间戳中减去创建议题的时间戳。如果议题的作者创建了响应，则不计算响应。

参考资料

议题解决时长

问题：解决议题花费了多长时间？

描述

这一指标表明议题在解决之前平均有多长时间处于未解决状态。议题如已解决议题中定义。

对于重新打开后再次关闭的议题，只有最后的解决日期与该指标有关。

目标

此指标可用于评估完成和解决讨论所需的工作量和时间。此指标还可以提供项目响应程度的相关信息。

实现

对于每个已关闭的议题：

- 议题解决时长 = 已关闭的议题时间戳 - 新建议题的时间戳

聚合器：

- 平均数。仓库中议题得到解决的平均时间（默认以天为单位）。

参数：

- 时间段。开始日期和完成日期。默认：永久。
  考虑议题的时间周期。

筛选条件

- 按时间。由提供的开始日期到提供的结束日期，表明平均议题解决时长。
  - 按议题建立时间。由提供的开始日期到提供的结束日期，在此时间段内新创建的议题是花多长时间解决的（该议题的解决时间可能晚于规定的时间段）。
  - 按议题关闭时间。由提供的开始日期到提供的结束日期，在此时间段内解决的议题是花多长时间解决的（该议题创建时间可能早于规定的时间段）。
- 按参与者（提交者、评论者、解决者）。需要参与者合并（合并同一作者的对应 ID）。
- 按参与者群组（雇主、性别...涉及每个参与者）。需要参与者分组，并且可能需要参与者合并。

可视化效果

- 一段时间的平均数（如一月平均数、二月平均数、三月平均数等）
- 给定时间段的平均数（例如，2019 年的全年平均数，或一月至三月的平均数）

提供指标的工具

- Augur 将此指标作为已解决议题解决时长提供。该指标均以 `repo` 和 `repo_group` 指标形式提供，更多信息见 Augur 文档。

数据收集策略

关于收集已解决议题的数据的具体说明，请参阅“已解决议题” 的对应部分。

参考资料

关注领域 - 社区成长

目标: 确定项目社区的规模，以及它是在增长、缩小还是保持不变。

度量指标	问题
贡献归属	谁对开源项目做出了贡献，对一个贡献相关个人和组织的归属信息是如何分配的?
不活跃贡献者	在特定时间段内有多少贡献者处于不活跃状态?
新贡献者	有多少贡献者为给定项目做出第一次贡献？他们是谁?
首次关闭议题的贡献者	在给定项目中，有多少贡献者是第一次关闭议题?

贡献归属

问题: 谁对开源项目做出了贡献，对一个贡献相关个人和组织的归属信息是如何分配的?

描述

这个指标评估谁参与了项目和具体的项目任务，并提供项目贡献者和所属组织的归属信息。其目的是通过洞察社区的有偿贡献动态，来了解 “工作是如何完成的”。

目标

1. 谁在做这个项目?
2. 志愿者工作、有偿工作和混合工作（志愿者与有偿工作混合）的比例是多少? What is the ratio of volunteer work, sponsored work, and blended work?
3. 有多少贡献是有偿做出的?
4. 谁赞助了这些贡献?
5. 哪些类型的贡献是被赞助的?
6. 在一个项目中，贡献者的社区多样性如何?

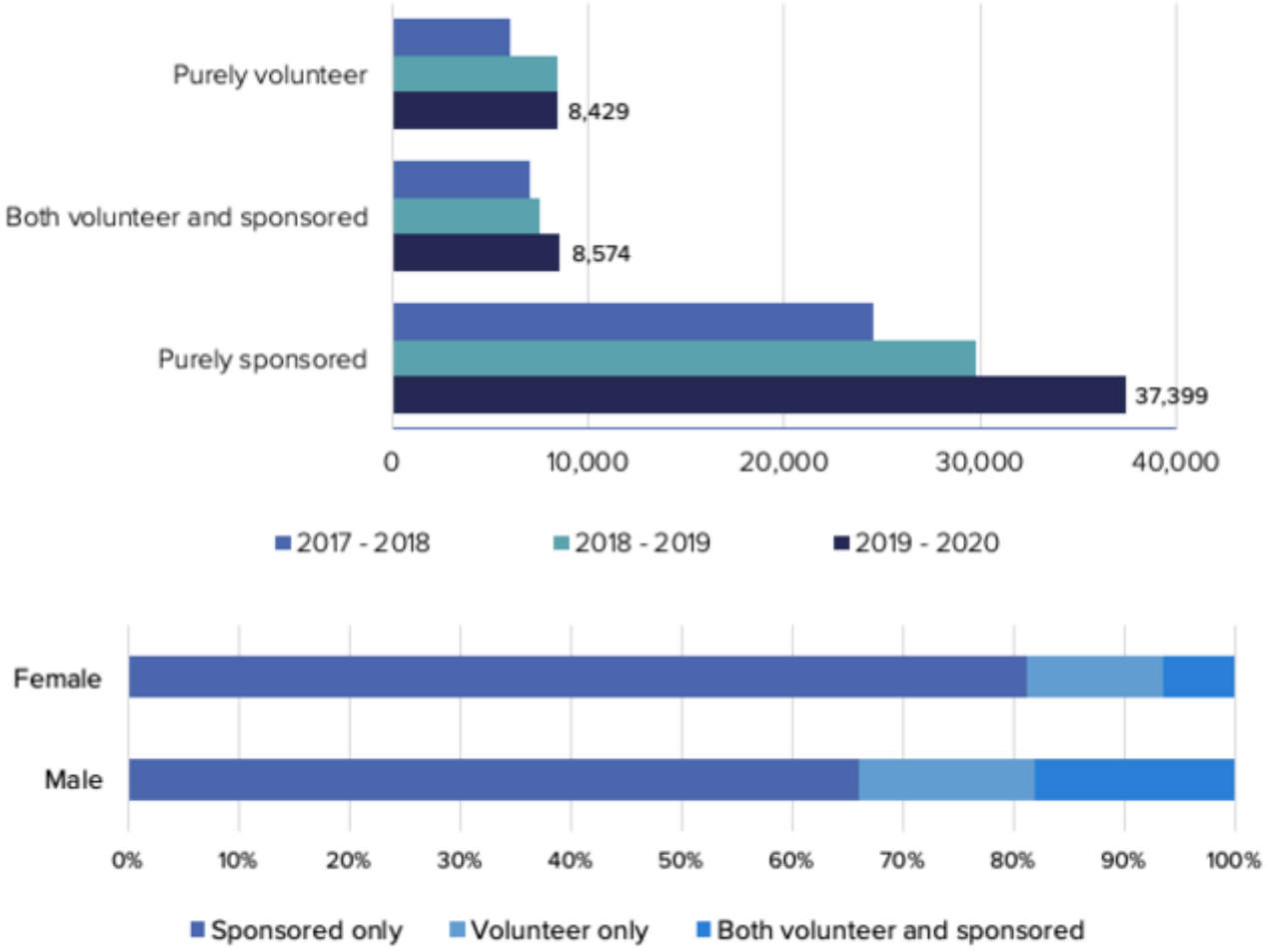
实现

大多数贡献都可以使用跟踪数据隐式地进行属性化，这些属性在其他度量中得到反映。然而，这个度量很大程度上依赖于贡献者自愿提供并由项目领导解释的数据。这个度量的实现要求循环中的人确定哪些组织以及哪些个人贡献者做出了贡献。每个独立的贡献者都应该有机会指出哪些公司、基金会、项目和/或客户为特定的贡献付费。

过滤条件

- 贡献者类型 (个人，组织，性别，种族，全球状态，工作地点)
    - 志愿者
    - 由公司和/或客户赞助
    - 贡献者在项目中扮演的角色(例如，维护者，董事会成员等)
- 贡献类型
    - 在相关的地方，链接到贡献交付件，如合并请求、议题等。
    - 指出 Git 平台中没有管理到的贡献类型，如 GitHub、GitLab 和 BitBucket。
    - 参见: https://chaoss.community/metric-types-of-contributions/
- 志愿者 vs 有偿贡献 - 涉及到 组织多元化与人力投资

可视化效果



提供指标的工具

1. Drupal 社区构建了这个工具，并在2015年开始使用, 并每年公布结果。
2. 在 Gitlab 有一个打开的议题来实现这个功能。

数据收集策略

The Drupal Community implemented an example of how to gather information necessary for this metric to be calculated. It associates individuals, and organizations those individuals indicate as warranting attribution, for each discrete contribution. Drupal 社区实现了一个例子，说明如何收集计算这个度量所必需的信息。它将个人和这些人表示需要归属的组织与每一个独立的贡献联系起来。

数据伦理的考虑

虽然这个指标需要捕捉个人和他们所做贡献之间的关系，但这个指标的目的不是为了衡量个人。我们的目的是让大家更广泛地了解对这个项目的贡献是如何被激励的。明确地说，本指标的目的不是要对个人贡献者的工作进行游戏化。

参考资料

- https://dri.es/a-method-for-giving-credit-to-organizations-that-contribute-code-to-open-source
- https://www.drupal.org/blog/who-sponsors-drupal-development
- https://dri.es/who-sponsors-drupal-development-2020

- https://gitlab.com/gitlab-org/gitlab/-/issues/327138

贡献者

- Matthew Tift
- Sean Goggins
- Elizabeth Barron
- Vinod Ahuja
- Armstrong Foundjem
- Kevin Lumbard

不活跃贡献者

问题：在特定时间段内有多少贡献者处于不活跃状态?

描述

该指标显示特定时间段内有多少贡献者停止了贡献。贡献者被记为不活跃所需的时间可以由一个变量决定，此指标将展示在给定时间框架内被标记为不活跃贡献者的数量。

目标

确定有多少人停止了活跃贡献。这可以让社区管理者高效确定关键成员是否将失去兴趣或准备休息。

实现

该指标将包含两个变量 - 时间段和时间间隔。时间段将是显示不活跃成员数量的时间段。例如，如果时间段 = 年，那么它将显示每年的不活跃贡献者数量。时间间隔将决定贡献者被标记为不活跃所设置的时间。如果贡献者没有做出贡献的时间超过时间间隔，即被视为不活跃。

指标运作方式：

1. 获取所有贡献者的列表
2. 检查最后的贡献日期
3. 如果最后一次贡献日期在截止日期之前，则将其添加到最后一次贡献所处时间段的不活跃计数中。
4. 创建不活跃贡献者列表

聚合器：

- 不活跃：不活跃贡献者数量

筛选条件

- 被视为活跃所需的最小贡献
- 确定不活跃的时间段
- 开始日期/结束日期
- 图形周期

数据收集策略

可以使用现有贡献者指标来收集贡献者列表。要确定最后的贡献日期，可能需要新的代码。

新贡献者

问题：有多少贡献者为给定项目做出第一次贡献？他们是谁？

描述

新贡献者的增加或减少可能是项目健康的早期指标。了解社区新成员的行为和障碍，需要知道他们是谁 - 用来帮助欢迎新贡献者并感谢他们的付出。

这是贡献者指标的特定实现。

目标

新贡献者的增加或减少可能是项目健康的早期指标。了解社区新成员的行为和障碍需要知道他们是谁。欢迎新贡献者，并感谢他们做出的第一次贡献。

实现

对每个贡献者，仅考虑其第一次贡献。

筛选条件

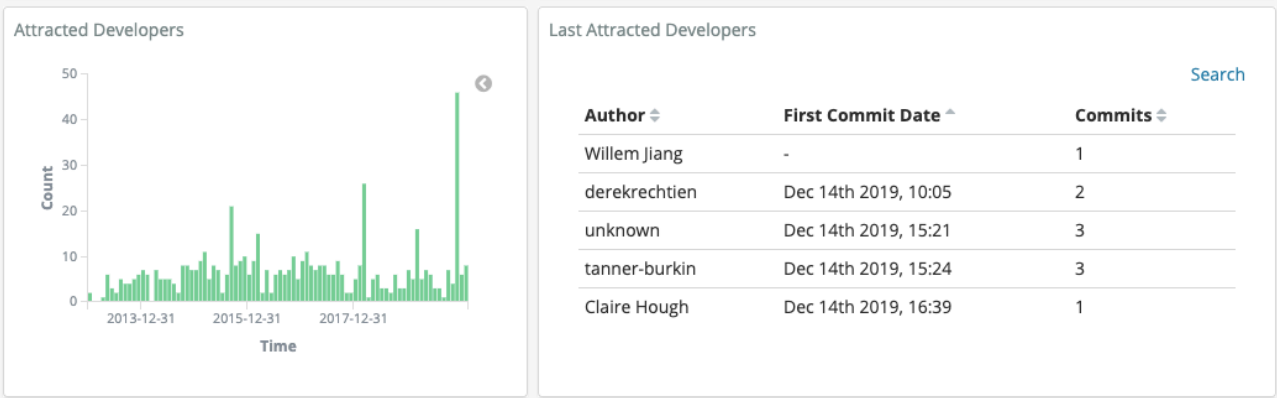
时间段：做出第一次贡献的时间。

按参与地点。例如：

- 仓库作者
- 问题作者
- 代码审查参与者
- 邮件列表作者
- 事件参与者
- IRC 作者
- 博客作者
- 按发布周期
- 项目活动的时间框架，例如，寻找新贡献者
- 项目的编程语言
- 项目中的角色或职能

参与阶段（例如，建立拉取请求与使其获得接受）。

可视化效果



提供度量的工具

Augur GrimoireLab

参考资料

https://opensource.com/article/17/4/encourage-new-contributors

首次关闭议题的贡献者

问题：在给定项目中，有多少贡献者是第一次关闭议题?

描述

该指标表示在给定项目中首次关闭议题的贡献者数量。当贡献者第一次关闭议题时，这表明该项目中的个人“粘性”，特别是贡献者不为代码提交者的情况。

目标

将“关闭议题”确定为贡献者旅程中的一个里程碑，了解贡献者如何在贡献者漏斗中移动。

实现

聚合器：

- 计数。在给定时间段内，首次关闭此项目议题的贡献者总数。
- 百分比。在给定时间段内，首次关闭此项目议题的贡献者占该时间段内关闭过议题的贡献者总数的比例。

参数：

- 时间段。计算首次问题关闭者的开始日期和结束日期。默认：永久（即整个可观察项目生命周期）

筛选条件

- 排除重新打开的议题（如果在不到 1 小时的时间内重新打开，则可以选择筛选条件）

可视化效果

- 表中列出首次关闭议题的贡献者的姓名和对应时间。
- 时间轴在 x 轴显示时间，y 轴为固定连续时间段（如每月）的汇总指标值。这种可视化效果可以显示所考虑项目的指标随时间的变化情况。

数据收集策略

基于已解决议题和贡献者定义，通过首次解决议题的日期完善贡献者信息。

参考资料

- 贡献者漏斗 Mike McQuaid

Risk WG

关注领域	目标
商业风险	了解一个社区如何围绕或支撑软件包开发。
代码质量	了解软件包的质量
依赖风险评估	了解软件的依赖风险。
许可证	了解与软件包使用相关的潜在的知识产权问题
安全	了解与软件开发相关的安全流程和程序

关注领域 - 商业风险

目标: 了解一个社区如何围绕或支撑软件包开发。

度量指标	问题
巴士系数	如果最活跃的人离开，项目会产生多大风险?
Committers	社区贡献者的健壮性如何?
大象系数	社区的工作分配是怎样的?

巴士系数

问题：如果最活跃的人离开，项目会产生多大风险？

描述

巴士系数是一个引人注目的指标，因为它形象化了“我们失去多少贡献者就会使项目停滞？”的问题。假设这些贡献者被一辆公共汽车撞了，（换个令人容易接受点的说法是，有多少人中了彩票并决定离开项目）。

巴士系数是贡献超过整个项目 50% 的最少人数。

目标

- 确定项目工作在贡献者之间分布的范围。
- 确定项目中负责大部分工作的关键人员。

实现

巴士系数的公式是一个百分比计算 - 这里我们使用 50% 作为阈值 - 然后将每个贡献者的贡献按降序排序，直到贡献总和到达门槛。

如果我们有 8 个贡献者，每个贡献者在项目中的贡献值如下：`1000, 202, 90, 33, 332, 343, 42, 433`，那么我们可以按照降序排序后从最大值开始求和，用第一次超过贡献总数的 50% 的贡献者个数来确定巴士系数。

答案: 贡献值的 50% = `1,237.5`，所以巴士系数为 `2` 。

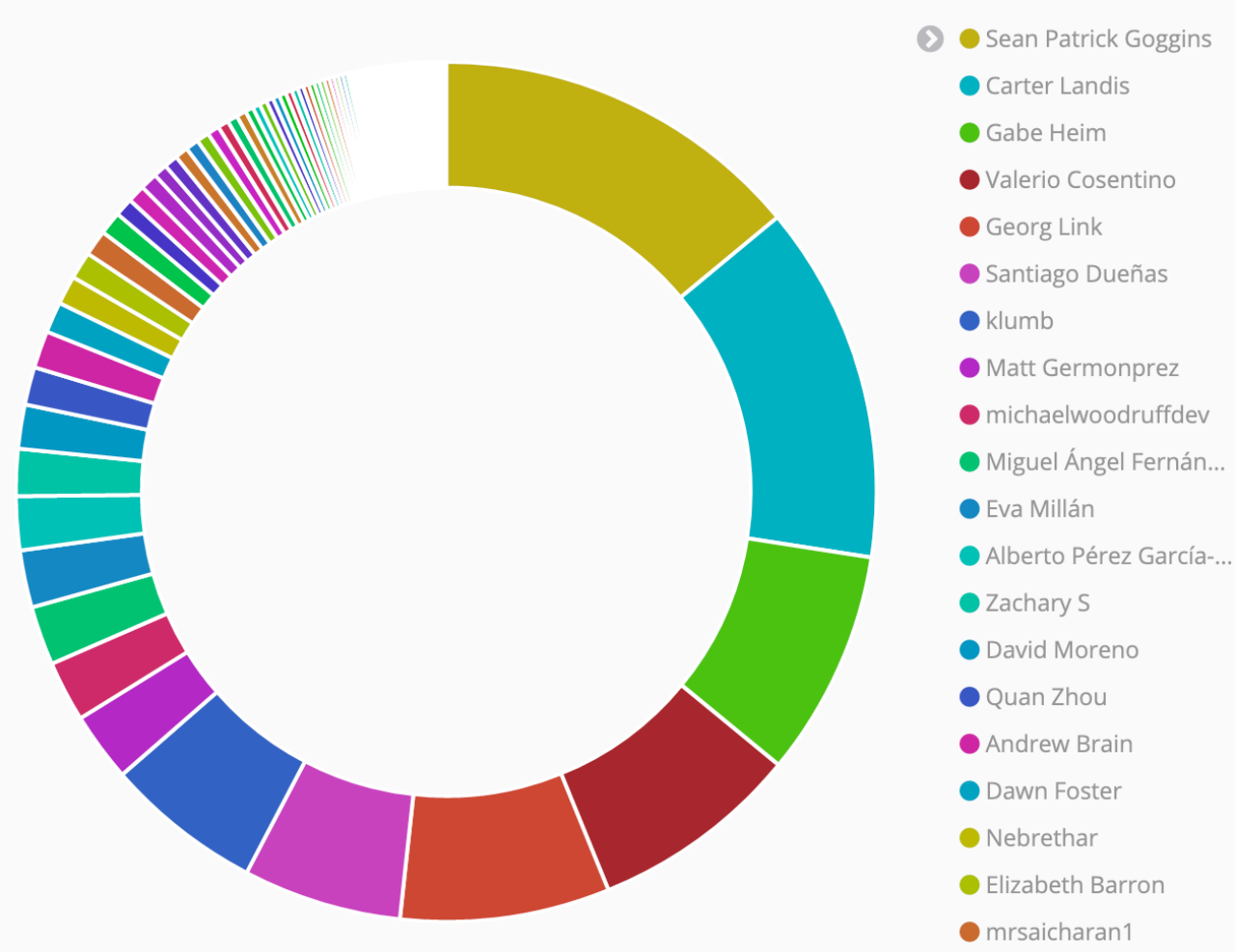
求解过程:

1. 按降序排列数据： `1000, 433, 343, 332, 202, 90, 42, 33`
2. 计算总数的 50%:
   - `(1,000 + 433 + 343 + 332 + 202 + 90 + 42 + 33)  0.5 = 1,237.5`
3. 将排名中的前 2 个贡献者相加，我们得到 `1,433` 。
4. **答案**：因为 `1,433 > 1,237.5` ，超过 50% 的贡献仅由 `2` 贡献者完成，因此 `巴士系数 = 2` 。

筛选条件

- 时间：巴士系数可能因不同的时间段而异。项目生命周期内的巴士系数可能会歪曲项目中贡献者的当前参与程度。
- 代码仓组：许多开源项目包含多个代码仓，在某些情况下，检查与任何给定项目关联的所有代码仓可提供更完整的巴士系数图像。

可视化 (可选项)



仅考虑 git commits 时，2020 年 CHAOSS 项目的巴士系数为 5。

提供指标的工具

1. Augur
2. GrimoireLab 开箱即用地提供此指标，它不仅仅只是一个数字，而是可视化的。

数据收集策略

数据收集策略依赖于用于计算巴士系数的贡献类型。巴士系数通常是以 commits 数来计算的，但其他类型的贡献也可以用于此目的，它可以单独计算或也可以用不同贡献类型组合计算。

参考资料

Committers

问题：社区贡献者的健壮性如何？

描述

Committers 指标是已向项目提交代码的个人数量。这与更广泛构造的“贡献者”CHAOSS 指标不同，直接针对管理者在决定使用哪个开源项目时，风险评估中出现的一个特定问题。虽然不一定在所有情况下都是如此，但人们普遍认为，一个项目的贡献者越多，该项目就越有可能继续得到更新、支持和必要的资源。因此，该指标使各组织能够在知情的情况下决定，给定项目的提交者人数是否可能在当前或未来构成项目被放弃或支持不足的风险。

目标

从管理者决定将开源项目纳入其组织的角度来看，提交者的数量有时很重要。特别是代码贡献与更大规模的贡献者指标（包括文档作者、创建问题的个人和其他类型的贡献）可能会有很大不同，具体取决于开源项目所采用的管理风格。McDonald 等人 (2014) 让人们注意到不同开源项目如何遵循开放分布式模型，而其他项目则遵循高度集中式模型。较少的代码贡献者可能表明项目对外部贡献的开放程度较低，或者是项目中只有少数人了解并对代码库做出贡献。

实现

在开源项目中，每个有提交合并到项目的电子邮件地址都是一个“提交者”（见下一节的“已知问题”）。建议确定特定时间段内唯一的提交者数量，这样做的公式也很简单：

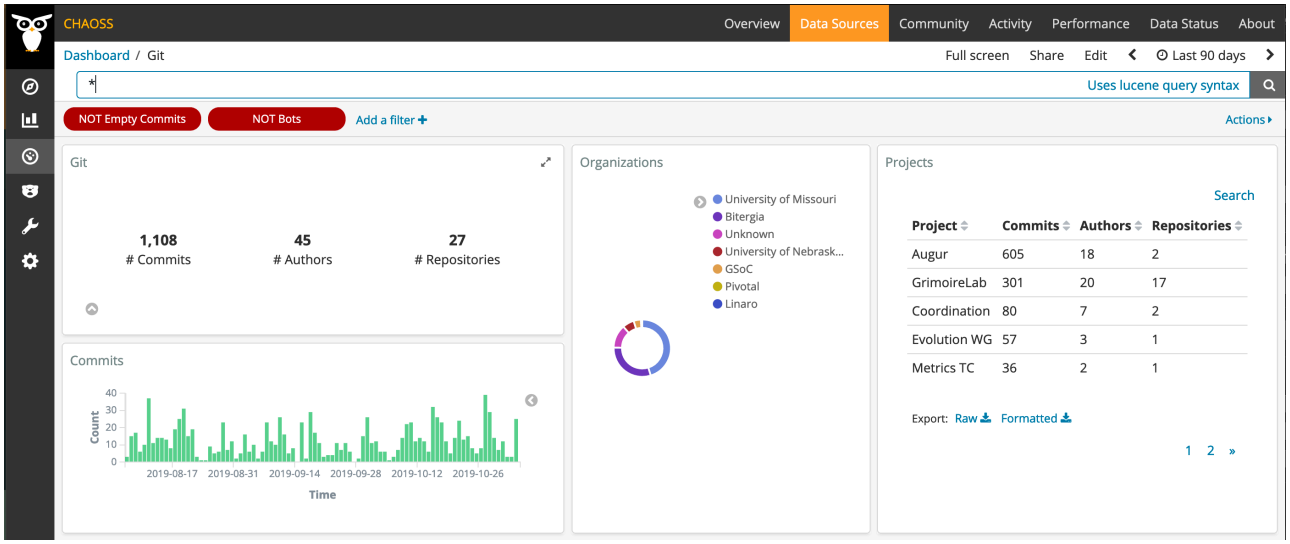
`Number_of_committers = distinct_contributor_ids (在具有开始日期和结束日期的一段时间内)`。例如，我可能想知道在过去的 18 个月里，有多少不同的人向一个项目提交了代码。`Committers` 揭晓答案。

数据质量的已知问题

- 许多贡献者使用不止一个邮箱，如果不协调这些共享身份，可能会人为地提高提交者总数。
- 几个提交者“顺便”做出一些微小的贡献，也可能人为地提高这一数字。

可视化效果

Grimoire Lab 显示提交者

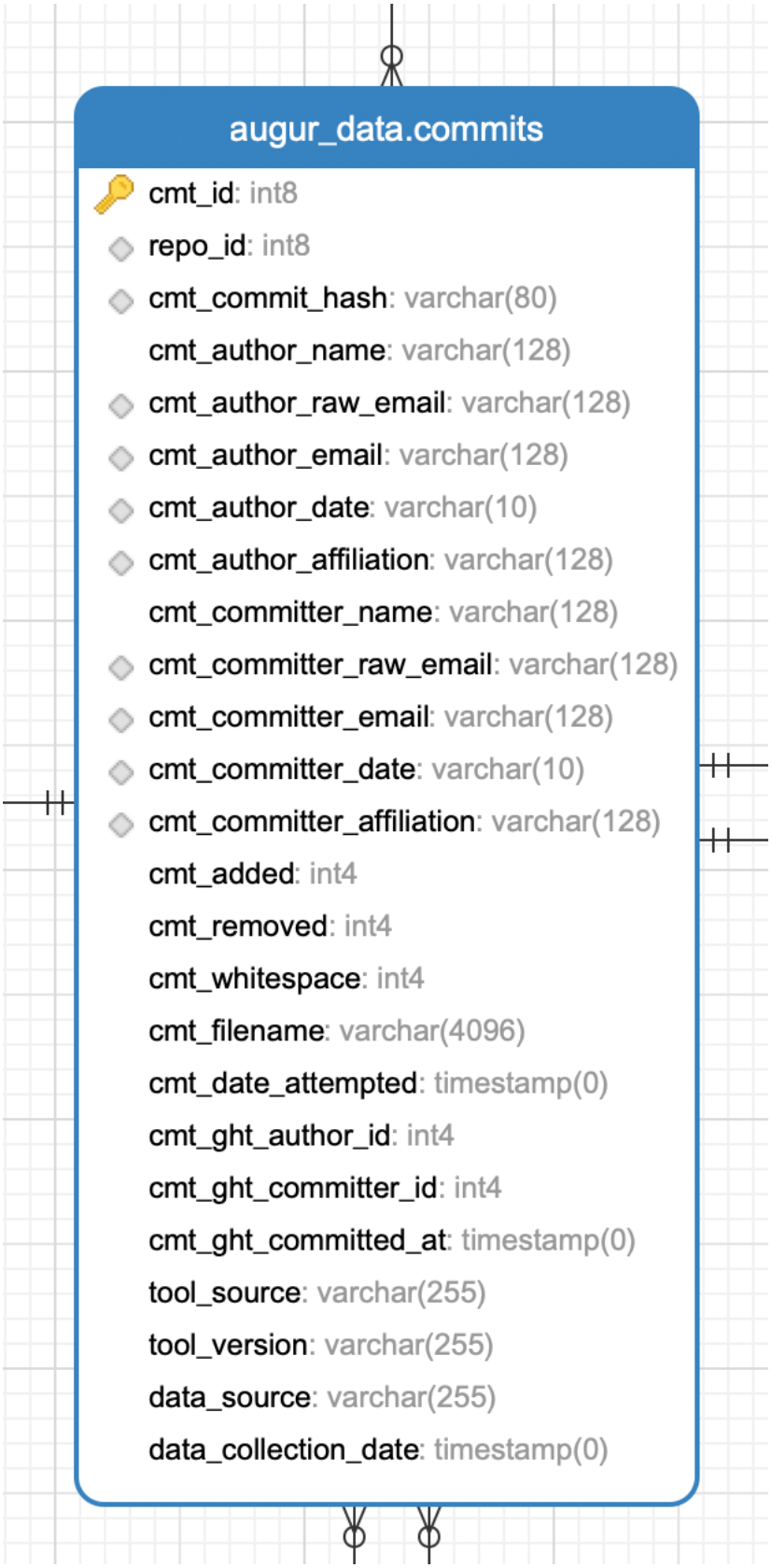


筛选条件

- 时间：较新的不同 Committer 的数量可能比项目（代码仓）整个生命周期内的人数更能清楚地表明参与项目的人数。
- 提交大小：按代码行数衡量的少量代码提交可以被排除，避免已知问题
- 提交次数：某一时间段内提交次数少于某个最低限度的贡献者可从这一数字中排除。

提供指标的工具

Augur 为代码仓中的每条提交记录维护了一个表格。



要评估一个代码仓的不同提交者，可以使用以下 SQL 或文档化的 API 端点：

该表达式使最终用户能够按提交计数阈值轻松筛选，返回的行数就是代码仓的 “Total_Committers”。

Grimoire Lab 还额外提供了针对 Committer 的洞察能力。

参考资料

1. Nora McDonald, Kelly Blincoe, Eva Petakovic, and Sean Goggins. 2014. Modeling Distributed Collaboration on GitHub. Advances in Complex Systems 17(7 & 8).

大象系数

问题：社区的工作分配是怎样的？

描述

公司员工对软件仓库中的总提交比例进行可参数化定义，这些公司的最低数量即为项目的“大象系数”。例如，一种常见筛选方法是，如果 50% 的提交是由 `n 家公司` 完成，即为大象系数。从最大的贡献组织开始，加总到参数化百分比，然后是下一个最大的组织，以此类推。例如一个项目有 8 个贡献组织，每个组织在项目中贡献了 12.5% 的提交，如果将大象系数参数化为 50%，那么大象系数为 “4”。如果其中一个组织在同样的场景中负责 50% 的提交，那么大象系数就是 “1”。

大象系数提供了一个简单易行的指标，表明执行一定比例（即 50%）工作的公司的最低数量。“大象系数” 一词的起源在文献中并没有明确划分，不过它可能源于 Venters 等人 (2014) 对软件可持续性作为关键非功能性软件需求的普遍认同。

目标

一家公司在评估开源软件产品时，可能会使用大象系数比较项目对一小部分企业贡献者的依赖程度。大象系数低的项目在直观上更容易被一个企业的决策串联到任何消费该工具的企业。对于 1000 个组织贡献的项目和可能只有 10 个组织贡献的项目，参数化筛选条件应该有合理区分。在一定的组织贡献量下（可能少于 1000 个组织），大象系数很可能不是软件采购的核心考虑因素，因为明智的管理者会使项目不易被少数参与者的决定所影响。此类阈值很大程度上取决于具体环境。

实现

大象系数的计算公式是百分比的计算 - 它将是我们的阈值 - 然后每个公司的贡献按降序相加，直到达到阈值。

如果我们有 8 个组织，每个组织为项目贡献的提交数量：`1000, 202, 90, 33, 332, 343, 42, 433`，那么我们可以首先得出所有公司总提交的 50% 来确定大象系数。

** 汇总：** 总贡献的 50% = `1,237.5`，因此大象系数为 `2`。

完整解答：

1. 降序排列数据：`1000, 433, 343, 332, 202, 90, 42, 33`
2. 计算总数的 50%：
   - `(1,000 + 433 + 343 + 332 + 202 + 90 + 42 + 33)  0.5 = 1,237.5`
3. 将排名前两位的公司相加，得到 `1,433`。
4. **回答**：对于 `1,433 > 1,237.5`，超过 50% 的贡献仅由 `2` 家公司完成，因此 `大象系数 = 2`。

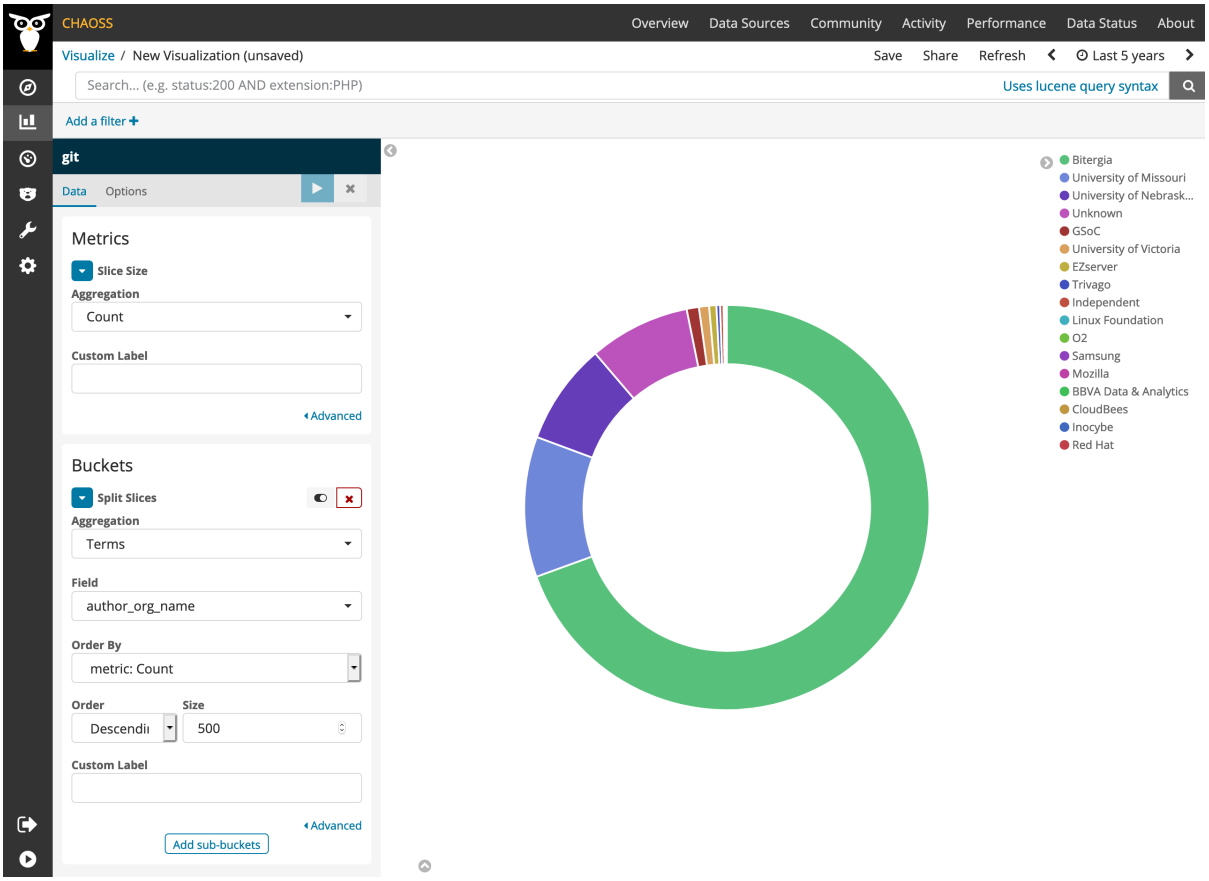
筛选条件

- 时间：如果先前时间段存在快照，大象系数就会适度地发生变化。因此，产品生命周期中的大象系数可能会错误地代表项目支持的当前组织多元化水平。
- 仓库组：许多开源项目包括多个仓库，在某些情况下，检查与给定项目相关的所有仓库可以更全面地了解大象系数。

提供指标的工具

1. Augur
2. GrimoireLab 提供此指标，开箱即用，非单一数字而是可视化效果。

- 查看 Bitergia Analytics 的 CHAOSS 实例示例。
- 从 GrimoireLab Sigils 面板集合下载并导入包含此指标可视化效果示例的现成仪表板。
- 按照说明向任意 GrimoreLab Kibiter 仪表板添加一个示例可视化效果：
  - 新建一个 `Pie` 图
  - 选择 `git` 索引
  - 指标切片大小：`Count` 聚合
  - 桶剪接切片：`Terms` 聚合，`author_org_name` 字段，`metric: Count` 排序依据，`Descending` 排列，`500` 大小
- 屏幕截图示例：



参考资料

1. Colin C. Venters, Lydia Lau, Michael K. Griffiths, Violeta Holmes, Rupert R. Ward, Caroline Jay, Charlie E. Dibsdale, and Jie Xu. 2014. The Blind Men and the Elephant: Towards an Empirical Evaluation Framework for Software Sustainability. Journal of Open Research Software 2, 1. https://doi.org/10.5334/jors.ao
2. http://philslade.blogspot.com/2015/07/what-is-elephant-factor.html
3. https://blog.bitergia.com/2016/06/07/landing-the-new-eclipse-open-analytics-dashboard/

4. https://www.stackalytics.com/

关注领域 - 代码质量

目标: 了解软件包的质量

度量指标	问题
测试覆盖率	代码的测试情况如何?

测试覆盖率

问题：代码的测试情况如何？

描述

测试覆盖率描述了至少一个测试套覆盖了多少给定的代码仓。测试覆盖率有两个主要的衡量标准。一个是针对代码仓运行的测试套中涵盖的**子例程**的百分比。测试覆盖率的另一个主要含义是测试套执行期间覆盖的**语句**的百分比。“测试覆盖率” 在 CHAOSS 中的度量定义包括这两个单独的度量指标。

编程语言将**子例程**特指为 “函数”、“方法”、“例程”，或在某些情况下称为 “子程序”。在此定义中，特定代码仓的覆盖率仅限于特定代码仓中定义的方法，并且不包括代码仓所依赖的库或其他软件的覆盖率。

目标

了解测试覆盖水平是代表软件质量的一个信号。测试覆盖率低的代码表明软件工程实践可能不太严格，并且相应地增加了在部署和使用时检测到缺陷的可能性。

实现

语句包括变量赋值、循环声明、对系统函数的调用、“go to” 语句以及函数或方法完成时常见的 `return` 语句，其中可能包括也可能不包括返回 `value` 或 `value 数组` 。

子例程覆盖率

$$subroutine-coverage-percentage = \frac{subroutines - tested}{total - subroutines - in - repository} * 100$$

语句覆盖率

$$statement-coverage-percentage = \frac{statements - excecuted - in - testing}{total - statements - in - repository} * 100$$

筛选条件

- 时间：按时间变化过滤测试覆盖率，可以帮助项目关注最大化整体测试覆盖率。具体参数包括时间段的 “开始日期” 和 “结束日期”。
- 代码文件: 每个代码仓都包含许多代码文件。按特定文件过滤覆盖率提供了更精细的测试覆盖率视图。某些函数或语句可能会导致比其他函数或语句更严重的软件故障。例如，安全关键系统的 “故障安全” 功能中未经测试的代码比 “字体颜色” 功能测试更重要。
- 编程语言：大多数当代开源软件代码仓都包含几种不同的编程语言。每个 `Code_File` 的覆盖率都可以按照编程语言进行过滤。

提供指标的工具

- 提供测试覆盖率信息
  - Python 的主要测试框架 PyTest
  - 用于 Python 的 Flask Web 框架支持覆盖测试

    - 可以从我的网站获取用于 Java、C 和 C++ 等常用语言的开源代码覆盖工具，包括这个。

- 存储测试覆盖率信息
    - Augur 将测试覆盖率展现为一个表格，该表是其代码仓中主代码仓表格的子项。每次计算测试覆盖率时，都会对每个测试的文件、测试使用的测试工具和文件中的语句/子程序的数量以及测试的语句和子程序的数量进行记录。通过在此粒度记录测试数据，Augur 启用了 “代码文件” 和 “编程语言” 汇总级别的统计信息和过滤器。

参考资料

1. J.H. Andrews, L.C. Briand, Y. Labiche, and A.S. Namin. 2006. Using Mutation Analysis for Assessing and Comparing Testing Coverage Criteria. IEEE Transactions on Software Engineering 32, 8: 608–624. https://doi.org/10.1109/TSE.2006.83

2. Phyllis G Frankl and Oleg Iakounenko. 1998. Further Empirical Studies of Test Effectiveness. In Proceedings of the 6thACM SIGSOFT International Symposium on Foundations of Software Engineering, 153–162.

3. Phyllis G Frankl and Stewart N Weiss. 1993. An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing. EEE Transactions on SoftwareEngineering 19, 8: 774–787.

4. Laura Inozemtseva and Reid Holmes. 2014. Coverage is not strongly correlated with test suite effectiveness. In Proceedings of the 36th International Conference on Software Engineering - ICSE 2014, 435–445. https://doi.org/10.1145/2568225.2568271

5. Akbar Siami Namin and James H. Andrews. 2009. The influence of size and coverage on test suite effectiveness. In Proceedings of the eighteenth international symposium on Software testing and analysis - ISSTA '09, 57. https://doi.org/10.1145/1572272.1572280

关注领域 - 依赖风险评估

目标: 了解软件的依赖风险。

度量指标	问题
依赖库年龄	与当前的稳定版本相比，项目的依赖库的年龄是多少?
上游代码依赖	我的项目依赖哪些项目和软件库?

依赖库年龄

问题: 与当前的稳定版本相比，项目的依赖库的年龄是多少?

描述

Libyears 解释了与项目依赖库的当前稳定版本相比，项目 (代码仓，或者可构建软件的代码仓) 依赖库的年龄是多少。依赖库的 “当前稳定” 版本是一个用于一般用途的版本，不包括候选/草稿版本。年龄是一个项目的所有依赖关系的累积，可以通过多种方式确定，比如报告总年龄、平均年龄，可能还有中位年龄 (注意，如果有长长的年龄尾巴，中位年龄可以隐藏这个问题)。一般来说，较低的依赖库年龄更好。工具还应该提供按年龄排序的依赖库列表，以提供更深入的了解。度量这一概念是在 “测量软件系统中的依赖新鲜度” 中由 Cox 等人 [Cox 2015] 提出的。

注意: 在某些情况下，使用旧版本代替当前版本可能是更明智的选择; 也就是说，这个度量提供了洞察以帮助其他人确定哪些旧版本在使用。

目标

依赖库年龄度量的目标是帮助识别对项目稳定性、安全性和脆弱性构成风险的依赖库。一个早已过时的组件更有可能具有公开知道的漏洞，而且也不太可能得到良好的支持。它是一个有用的初始过滤器，用于对依赖关系进行分类，以帮助识别最值得仔细检查的项目的依赖关系。

实现

参数

默认情况下，这些信息将存在于一个软件生态系统中 (例如 JavaScript 或 Maven) ，因为它们更容易计算。这个度量可以跨多个生态系统中计算，但是往往需要更多的信息，例如，跨生态系统的软件物料清单 (SBOM)。

此信息可以只考虑直接依赖关系，或者还可以包括所有可传递的依赖关系。如果包含传递性依赖关系，则更有可能发现潜在的问题，但并非所有这类工具都支持这一点。

如果一个项目有多个稳定/受支持的分支，那么在任何分支中考虑 “当前” 是否可以接受? 默认情况下，只考虑最新的稳定分支，因为随着时间的推移，通常较旧的分支得到的维护较少。另一种选择 (虽然不是默认选择) 是，如果旧版本获得积极支持，那么也可以接受旧版本为 “当前” 版本; 报告必须在使用该选择时明确注明。是否应该有一个宽限期，新的依赖关系仍然被视为” 当前”? 默认情况下，答案是否定的; 任何特定的宽限期都是任意的，关键是要尽量保持与时俱进。

过滤条件

依赖级别 (仅直接，包括传递依赖，等。定义在上游代码依赖度量指标里。

- 累积年龄
- 依赖的平均年龄
- 依赖年龄的中位数
- 依赖关系的排序列表 (按依赖库年龄降序排列) ，因此首先识别 “由于年龄而造成的风险最大的” 依赖关系

可视化效果

这是依赖库年龄的一个示例，以直接依赖项的累积度量例，在本例中，累积值为 103.78 依赖库年龄。

```
libyear -r requirements.txt
+-------------------------+-----------------+----------------+-----------------+
|         Library         | Current Version | Latest Version | Libyears behind |
+-------------------------+-----------------+----------------+-----------------+
|           pytz          |      2015.2     |     2019.3     |       4.54      |
|         urllib3         |      1.15.1     |     1.25.7     |       3.58      |
|         astroid         |      1.5.3      |     2.3.3      |       2.43      |
|          django         |     1.11.23     |      3.0       |       0.34      |
|      django-celery      |      3.2.1      |     3.3.1      |       2.54      |
|        httpretty        |      0.8.3      |     0.9.7      |       5.31      |
|         Pygments        |       1.6       |     2.5.2      |       6.81      |
|          flake8         |      3.6.0      |     3.7.9      |       1.01      |
|      django-waffle      |      0.14.0     |     0.18.0     |       1.66      |
|    requests_oauthlib    |      0.8.0      |     1.3.0      |       2.72      |
|   django-debug-toolbar  |       1.8       |      2.1       |       2.52      |
|         libsass         |      0.13.3     |     0.19.4     |       2.06      |
|     django-storages     |      1.6.6      |      1.8       |       1.65      |
|      edx-i18n-tools     |      0.4.2      |     0.5.0      |       2.02      |
|           six           |      1.10.0     |     1.13.0     |       4.08      |
|   djangorestframework   |      3.6.3      |     3.11.0     |       2.58      |
|          isort          |      4.2.15     |     4.3.21     |       2.05      |
|         futures         |      2.1.6      |     3.3.0      |       5.5       |
|          Pillow         |      2.7.0      |     6.2.1      |       4.8       |
| edx-django-release-util |      0.3.1      |     0.3.2      |       2.44      |
|      beautifulsoup4     |      4.6.0      |     4.8.1      |       2.42      |
|       mysqlclient       |   1.4.2.post1   |     1.4.6      |       0.77      |
|         newrelic        |    4.14.0.115   |   5.4.0.132    |       0.78      |
|          redis          |      2.10.6     |     3.3.11     |       2.16      |
|         oauthlib        |      2.1.0      |     3.1.0      |       1.21      |
|        django-ses       |      0.7.1      |     0.8.13     |       3.65      |
|           mock          |      1.3.0      |     3.0.5      |       3.79      |
|      django-hamlpy      |      1.1.1      |      1.2       |       1.52      |
|          bottle         |      0.12.9     |    0.12.18     |       4.1       |
|      pylint-django      |      0.7.2      |     2.0.13     |       3.44      |
|       user-agents       |      1.1.0      |      2.0       |       2.13      |
|          jsmin          |      2.2.1      |     2.2.2      |       1.15      |
|         Markdown        |       2.4       |     3.1.1      |       5.26      |
|         gunicorn        |      0.17.4     |     20.0.4     |       6.59      |
|         requests        |      2.18.4     |     2.22.0     |       1.75      |
|          pylint         |      1.7.2      |     2.4.4      |       2.39      |
+-------------------------+-----------------+----------------+-----------------+
Your system is 103.78 libyears behind
```

这张图片来源于 https://github.com/nasirhjafri/libyear

提供度量的工具

请注意，一些工具还可以计算版本号之间的差异 (例如，1.1.1 vs. 1.2.3)；这样信息量更大，但并非所有依赖关系都使用相同的版本编号方法，因此为了简单起见，我们将重点放在度量时间上。

下面是一些实现依赖库年龄工具的例子:

- https://github.com/nasirhjafri/libyear - 利用 requirements.txt 度量 Python。
- https://github.com/sesh/piprot
- https://github.com/chaoss/augur - deps_libyear_worker - 现在可以度量 Python 和 Javascript。
- https://github.com/jaredbeck/libyear-bundler - 利用 Gemfile 度量 Ruby。

参考资料

- [Cox 2015] “测量软件系统中的依赖新鲜度” by Joel Cox, Eric Bouwers, Marko van Eekelen, and Joost Visser, https://ericbouwers.github.io/papers/icse15.pdf
- https://libyear.com/

贡献者

- Sophia Vargas (Google)
- David A. Wheeler (Linux Foundation)
- Vinod Ahuja (University of Nebraska Omaha)
- Kate Stewart (Linux Foundation)
- Duane O’Brien
- Sean Goggins (University of Missouri / CHAOSS Project)

上游代码依赖

问题: 我的项目依赖哪些项目和软件库?

描述

这个度量的目的是了解一个开源软件中代码的依赖项的数量和类型。这个度量明确地排除了关于基础设施的依赖关系，比如数据库和操作系统，它们将作为一个独特的度量标准来开发。通过扩展，对上游代码依赖性的了解使项目能够使用其他 CHAOSS 指标来评估每个依赖性的健康性和可持续性。

目标

上游代码依赖度量指标旨在理解构建、测试或运行一个软件所需代码的依赖关系。上游代码依赖度量指标可以帮助识我的项目直接或者间接依赖哪些项目、软件库或者版本。

实现

上游代码依赖度量指标可以通过分析项目的依赖文件或使用现有的工具来实现，这些工具可以检查所使用编程语言的包管理器的数据 (例如，JavaScript npm 使用 package.json，Python 使用 pyproject.toml/requirements.txt，Ruby 使用 Gemfile/Gemfile.lock 等)。注意: C/C++ 通常使用系统包管理器。对于多编程语言混合的项目，情况变得更加复杂，因为需要扫描几个特定于语言的依赖文件。

参数

所有枚举的依赖项都应该包括该依赖项的特定版本。请注意，有些系统不支持或不使用 “版本锁定”，因此不会使用特定的版本。

- 依赖树的深度
    - 直接依赖 - 第一顺序的依赖，如在源代码和/或软件包管理器配置中所声明的（例如，requirements.txt，Gemfile 等）。
    - 可传递依赖 - 间接依赖性，也就是说，超出第一顺序依也称为嵌套或第二顺序依赖。例如，正在评估的项目 a 依赖于项目 b，项目 b 依赖于项目 c。对于项目 a，项目 c 是可传递依赖。
    - 循环依赖 - 自己最终依赖自己。在允许循环依赖的系统中，我们假设给定的依赖在这种情况下只计算一次。
- 依赖状态
    - 静态依赖 - 依赖存在于所有的情况中。
    - 动态依赖 - 在使用过程中和其他上下文中的依赖发生变化。
- 依赖外部服务，比如使用 API
- 执行依赖 - 执行软件所需的依赖性。请注意，某些类型的依赖项通常被排除在计数之外，如下所述。这些可能是下列一项或多项:
    - 构建依赖 - 构建一段软件所需的代码
    - 测试依赖 - 测试一段软件所需的代码
    - 运行依赖 - 运行一段软件所需的代码

- 编程语言运行时依赖细节 (例如，Python 的运行环境)？(预设值为 no)。之所以提供这些细节，是因为在安全关键系统中，运行时的依赖关系对于质量保证非常重要。
    - 通常情况下，决定哪种编程语言运行是由虚拟环境控制，例如 Python 中的 venv；在 Ruby 中，你通常会使用rbenv 或者 rvm来实现（& 通常包含在”Gemfile ” 或”Gemfile.lock ” 和.ruby-version 中）。
    - PyPi 正在稳步增加它的 “拒绝编译不兼容的库/依赖” 的逻辑，它开始 “破坏构建”。
    - 不幸的是，并非所有的打包系统都有一个记录所有可传递依赖项的版本信息的约定，即使是在它们的生态系统中 (从长远来看也应该记录)。
    - 在某些系统中，有许多可能的运行时（runtime）可能难以区分。(例如，Common Lisp 有很多实现，通常任何一个都可以。)
- 语言在 count 中的内置库 (例如，Python 中的 “ re”) ? (默认值为 no)
    - 通常，许多内置库都是可执行依赖项。然而，它们通常是通过选择编程语言实现时 “大量” 安装的，并且常常被排除在计数之外，以简化分析。
    - 例如: 默认情况下，`pip freeze` 不包括这些 “包含在语言中” 的软件库/依赖项。
- 相同依赖项的多个版本是独立计算的。有些系统支持系统内同一依赖关系的多个版本; 在这种情况下，它们是分开计算的。

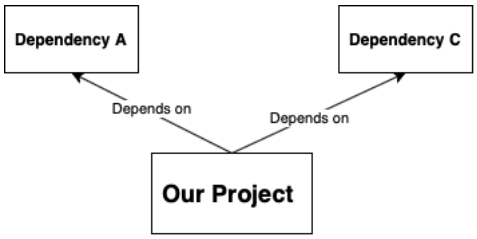
注意: 在运行时提供关于编程语言实现的主版本和次版本信息通常很重要。

- 一些计数和分析需要这些信息。通常忽略编程语言的运行时库（runtime libraries）和内置库 (见上文)，这些信息可以作为提供这些附加信息的简写。
- 示例: Ruby 生态系统在 Gemfiles 和.Ruby-version 文件中支持编程语言运行时的版本规范。
- 示例: PyPi 和 Anaconda 的 Python 版本经常以不同的方式管理不同版本的软件库。

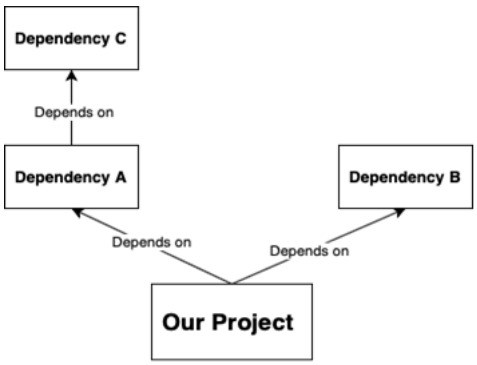
过滤条件

- 随时间变化的趋势 (例如，我依赖的项目比去年多还是少)
- 每个依赖项的版本数
- 引用相同依赖项的数量

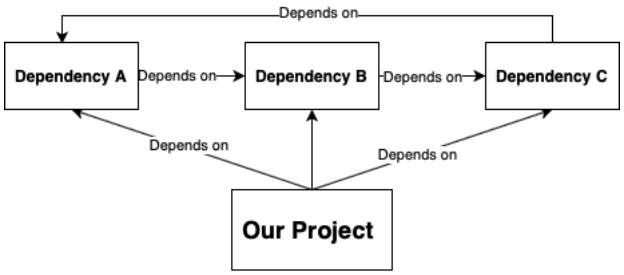
可视化效果



Direct Dependencies



Transitive Dependencies



Circular Dependencies

提供度量的工具

- deps.dev
- Deps.cloud
- OWASP
- Libraries.io
- BackYourStack.com
- Software bill of materials
- PackagePhobia

数据收集策略

- Augur 有一个代码扫描和包管理器扫描依赖标识的实现。
- Libraries.io 提供一个包管理器专注于依赖扫描 (也可以通过 Tidelift 使用).

参考资料

贡献者

- Georg Link
- Matt Germonprez
- Sean Goggins
- Sophia Vargas
- Kate Stewart

- Vinod Ahuja
- David A. Wheeler
- Arfon Smith
- Elizabeth Barron
- Ritik Malik
- Dhruv Sachdev
- Daune O’Brien
- Michael Scovetta

关注领域 - 许可证

目标: 了解与软件包使用相关的潜在的知识产权问题

度量指标	问题
许可证覆盖	代码仓有多少已声明许可证?
声明的许可证	什么是声明的软件包许可证?
OSI 批准的许可证	在项目的许可证中，OSI 批准的开源许可证所占比例是多少?
SPDX 文档	软件包是否具有关联的 SPDX 文档作为依赖项、许可证和安全相关问题的标准表达式?

许可证覆盖

问题：代码仓有多少已声明许可证?

描述

代码仓有多少扫描工具可以识别的已声明许可证，许可证可能不仅是 OSI 批准的。这包括软件和文档源文件，以总覆盖率的百分比表示。

目标

通过许可证覆盖率，可以深入了解软件包中具有已声明许可证的文件的百分比，从而得出两个用例：

1. 软件包供内部组织使用，已声明许可证覆盖可以突出使用该软件包时的利益点或关注点。
2. 此外，一个软件包提供到外部、下游项目，已声明许可证覆盖可以使下游集成、部署和使用所需的许可证信息透明化。

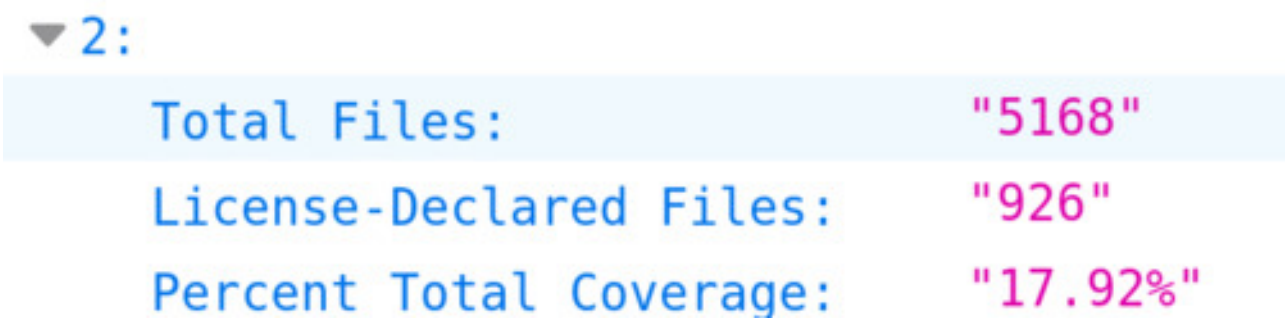
实现

筛选条件

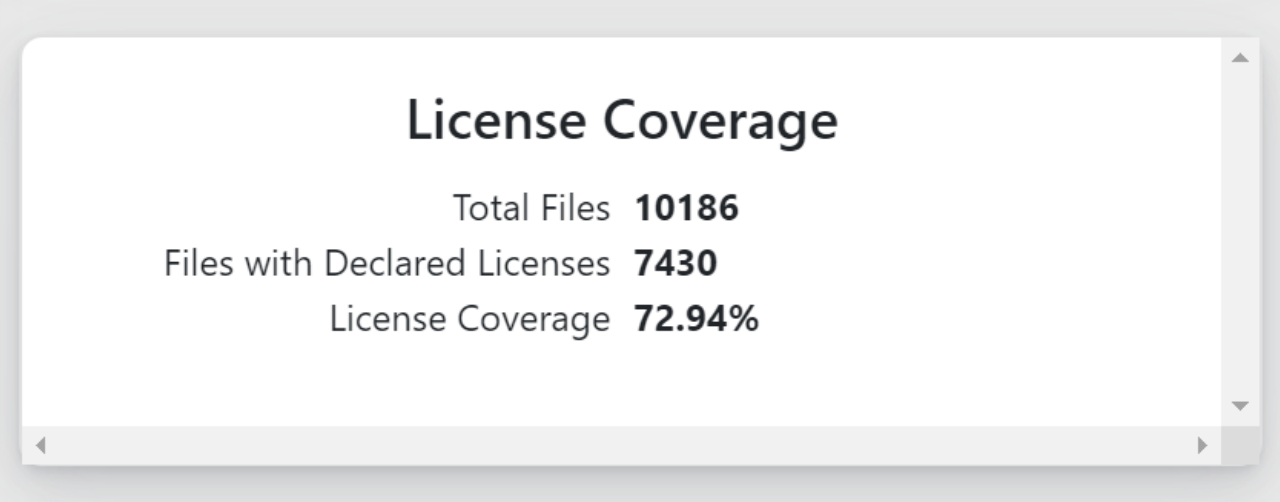
时间：代码仓依赖项随时间变化后，可能导致代码仓中声明的许可证也发生变化。除了基本了解之外，跟踪许可证存在的主要动机之一是注意意外的新许可证引入。

可视化效果

Augur 输出的 Web 演示：



Augur 输出的 JSON 演示：



提供指标的工具

1. Augur

可以提取和筛选数据以获得所需信息。可在任意 Augur 风险页面找到许可证覆盖数据。

参考资料

- https://spdx.org/
- https://www.fossology.org

声明的许可证

问题：什么是声明的软件包许可证？

描述

软件包中声明的特定许可证及其数量。它涵盖了软件源代码和文档源文件。此指标是许可证的枚举，以及带有该特定许可证声明的文件数。例如：

SPDX 许可证类型	带有许可证的文件数
MIT	44
AGPL	2
Apache	88

目标

声明的特定许可证及其数量在以下几种场景中至关重要：

1. 软件包的不变性使单个软件包可承载多个软件许可证，在获取软件包时，出于合规性考虑，必须了解声明的许可证。声明的许可证可以为许可证合规性工作提供透明度。
2. 许可证之间可能会产生冲突，使软件包中的所有许可证不能完全履行所有义务。声明的许可证可以为软件包中存在的潜在许可证冲突提供透明度。

实现

筛选条件

- 时间：代码仓依赖项随时间变化，可能导致代码仓中声明的许可证也发生变化。除了掌握已有许可证的基本信息外之外，跟踪许可证存在的主要动机之一是关注意外的新许可证引入。
- 已声明和未声明：单独枚举具有许可证声明的文件和没有许可证声明的文件。

提供指标的工具

1. Augur

可在任意 Augur Risk 页面的 “声明的许可证” 下找到声明的许可证。

1. Augur-SPDX

Augur-SPDX 包以 Augur 插件的形式提供，并使用这一数据模型存储文件级许可证信息。具体而言：

- 每个 `package` （repository）都可以有一个声明的或未声明的许可证，由代码仓中所有文件的扫描决定。
- 每个 `package` 还可以具有多个不同的非代码 `documents` ，这其中是 SPDX 许可证声明。

- 每个 `file` 可以与一个或多个 `packages_files` 关联。通过 `files` 和 `packages_files` 之间的关系，Augur-SPDX 让大量代码仓中的一个文件可以成为多个包的一部分，尽管在实践中这似乎不太可行。
- `packages` 和 `packages_files` 在两个方向上都具有一对多的关系。本质上，这让每个 `file` 都更有可能成为多个 `package` 的一部分，同样的，每个 `package` 通常都将包含许多 `package_files` 。
- `licenses` 与 `files` 和 `packages_files` 关联。每个 `file` 都可能具有不止一个 `licenses` 引用，例如 `license` 声明在 `Augur-SPDX` 每次扫描代码仓之间均可能发生变化。每个 `package` 均按照最近一次进行存储，每个 `packages_file` 都可以有一个 `license` 声明。

参考资料

- https://spdx.org/
- https://www.fossology.org

OSI 批准的许可证

问题：在项目的许可证中，OSI 批准的开源许可证所占比例是多少?

描述

这个指标提供了项目内所用开源许可证的透明度。需要透明的原因是一些许可证的传播实际上对开源并不友好。开源项目可能不希望包含未经 OSI 批准的许可证。

如 OSI 所述：“开源许可证是符合开源定义的许可证 - 简而言之，其允许软件被自由使用、修改和共享。要获得开源倡议（也称 OSI）批准，许可证必须经过开源倡议的许可证审查流程。”

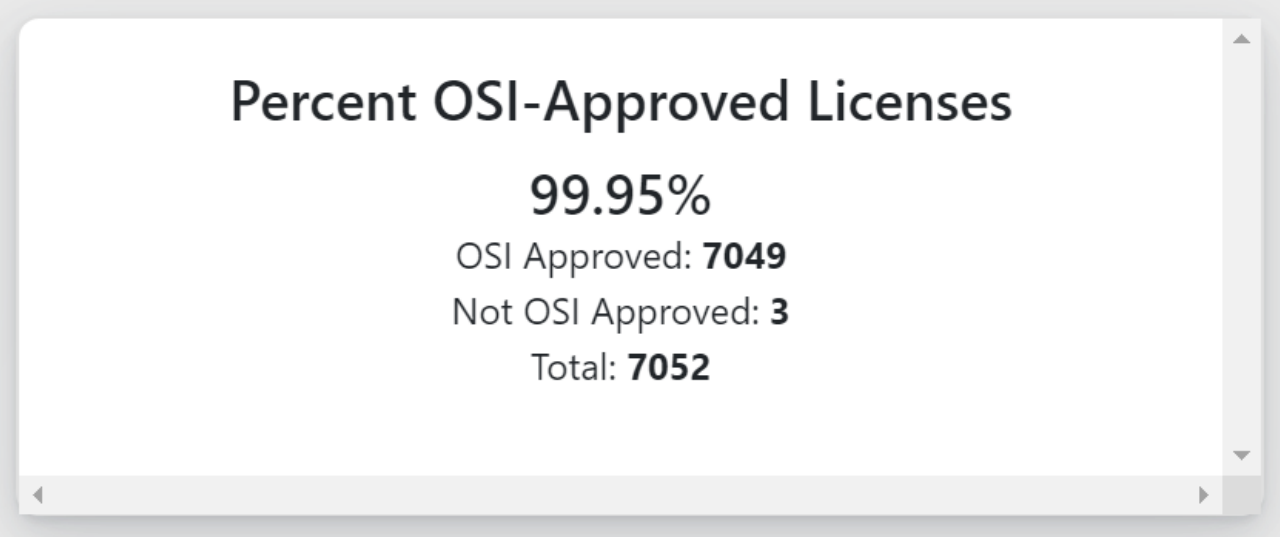
目标

确定项目是否含有不符合开源定义的许可证。这种透明度有助于项目有意识地决定是否包含未经 OSI 批准的许可证。

实现

可在 SPDX 提供的 Licenses.json 中找到 OSI 批准的许可证。

可视化效果



提供指标的工具

Augur 在项目的 “风险” 页面下提供该指标。

示例：http://augur.osshealth.io/repo/Zephyr-RTOS/zephyr/risk

数据收集策略

从代码仓提取许可证列表，与许可证覆盖率指标相同。将许可证列表与 Licenses.json 比较，记录 OSI 批准的许可证数量。计算 OSI 许可证列表上文件的百分比。

资源

- OSI 许可证页面
- SPDX 许可证列表

SPDX 文档

问题：软件包是否具有关联的 SPDX 文档作为依赖项、许可证和安全相关问题的标准表达式?

描述

软件包具有关联的 SPDX 文档作为依赖项、许可证和安全相关问题的标准表达式。有关 SPDX 规范的更多信息，请访问：https://spdx.org/

目标

对于获取开源软件用作 IT 或开源计划办公室组合的一部分的管理者，SPDX 文档提供的核心管理信息越发重要。这是因为软件包存在于复杂的软件供应链中，必须以标准化的方式明确表达该软件包的相关依赖项、许可证和安全相关问题。SPDX 文档为软件包的内部使用和下游分发提供了单一信息来源。SPDX 文档可帮助组织将开源工作常规化，以便更好地与其开源风险管理例程相结合。

实现

筛选条件

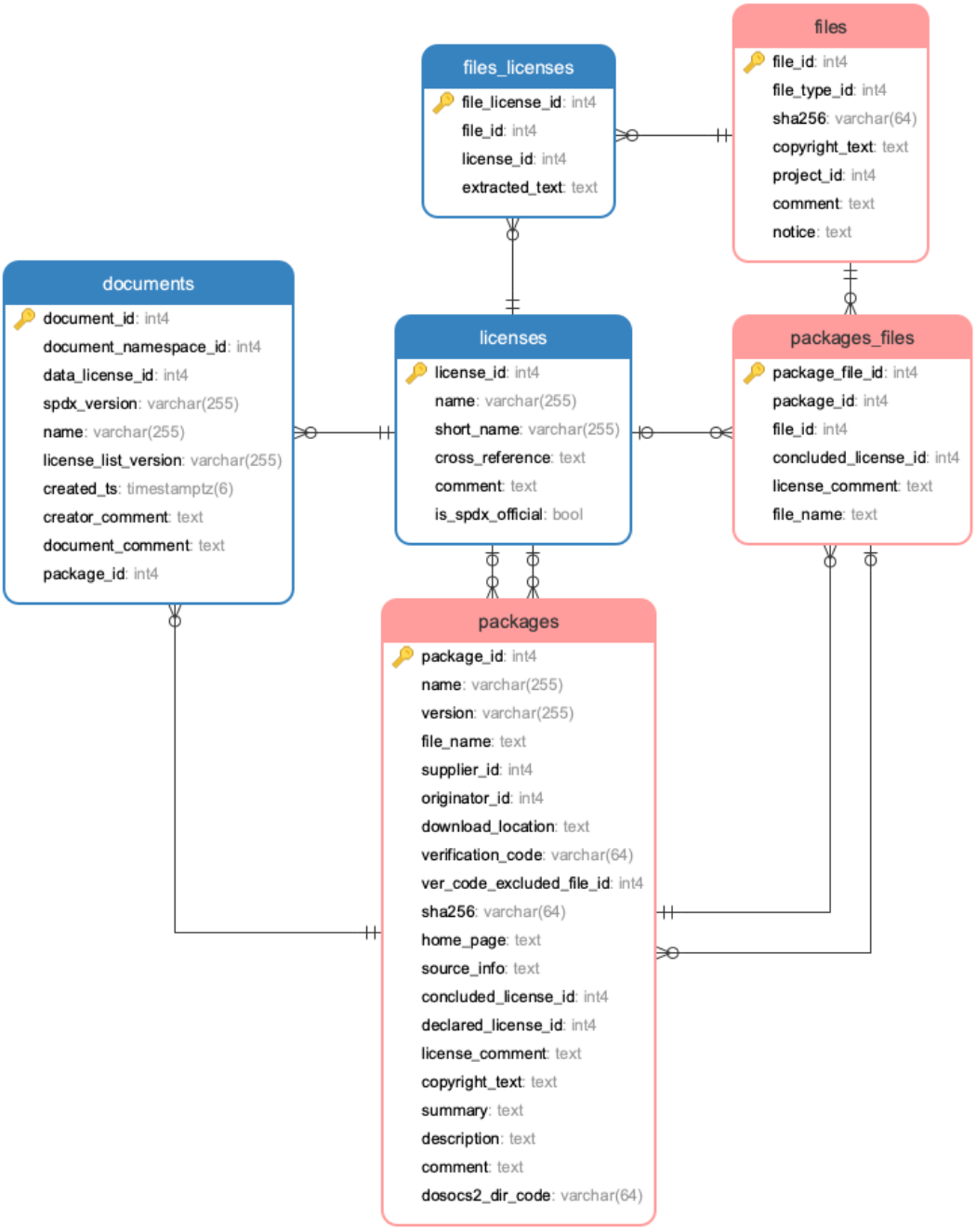
augur-SPDX 用于扫描 GitHub 代码仓 Zephyr。以下是从扫描中以 JSON 格式识别的许可证：

```
{
  "0": "Apache-2.0",
  "1": "BSD-2-Clause",
  "2": "BSD-3-Clause",
  "3": "GPL-2.0",
  "4": "GPL-2.0+",
  "5": "GPL-3.0+",
  "6": "ISC",
  "7": "MIT"
  "8": "BSD-4-Clause-UC",
  "9": "CC0-1.0"
}
```

本文档由 Augur 生成。

提供指标的工具

- DoSOCSv2，嵌入为 Augur 服务。使用 DoSOCSv2 插件配置的 Augur，可提供逐个文件的 SPDX 文档。数据库模式的相关部分如下所示。
- Augur-SPDX 嵌入为 Augur 服务。使用 augur-spdx 插件（该插件衍生自 DOSOCS）配置的 Augur，可提供逐个文件的 SPDX 文档。数据库模式的相关部分如下所示。此实现是 DoSOCSv2 的分支。
- 包
- Package_Files
- 文件（有可能也包含在其他包中，但可能性不大）。许可证信息被添加为 SBOM 的一部分，但许可证鉴别的复杂性详述于 License_Count、License_Coverage 和 License_Declared 指标。



参考资料

- https://spdx.org
- https://www.ntia.doc.gov/SoftwareTransparency

关注领域 - 安全

目标: 了解与软件开发相关的安全流程和程序

度量指标	问题
核心基础架构倡议：最佳实践徽章	项目目前的 CII 最佳实践状态如何？

核心基础架构倡议：最佳实践徽章

问题：项目目前的 CII 最佳实践状态如何？

描述

如 CII 最佳实践徽章页面所述：Linux 基金会核心基础架构倡议 (CII) 最佳实践徽章是开源项目展示其遵循最佳实践的一种方式。项目可以使用此 Web 应用程序解释其如何遵循每种最佳实践，从而免费自愿地进行自我认证。如果项目满足所有必要标准，即可获得合格徽章。

目标

如 CII 最佳实践徽章页面所述：CII 徽章表示项目对 Linux 基金会核心基础架构倡议定义的 “开源项目最佳实践” 的遵循程度，该倡议重点关注开源软件的网络安全。该计划的目标是鼓励项目生产更好、更安全的软件，并允许其他人确定项目是否遵循最佳实践。

徽章的消费者可以快速评估哪些开源项目遵循了最佳实践，从而更有可能生产出更高质量的安全软件。

实现

相关信息见 CII 的 API 文档。

可视化效果

提供指标的工具

Augur 为 CII 最佳实践指标提供了示例实现。使用的 CII 示例见 http://augur.osshealth.io/repo/Zephyr-RTOS/zephyr/risk

数据收集策略

相关信息见 CII 的 API 文档。

如 CII 最佳实践徽章页面所述：如果项目满足所有必要标准，即可获得合格徽章。对于给定项目，每个标准的状态可以是 “满足”、“未满足”、“不适用” 或 “未知”。每项标准都有四个类别：“必须”、“应该”、“建议” 或 “未来”。要获得徽章，所有 “必须” 和 “绝不” 标准必须满足，所有 “应该” 标准必须满足或者记录有不执行该标准的理由，所有 “建议” 标准必须被评价为满足或未满足。如果项目满足附加标准，可获得银级和金级的高级徽章。

参考资料

- CII 徽章网站：https://bestpractices.coreinfrastructure.org/en
- Augur：https://github.com/chaoss/augur

Value WG

关注领域	目标
学术价值	确定项目对研究人员和学术机构的价值程度。
公共价值	确定项目 (包括下游项目) 是否对社区用户或者贡献者有价值。
个人价值	识别一个项目对我作为个人用户或贡献者是否有价值
人力投资	从组织的角度看项目是否具有经济价值。

关注领域 - 学术价值

目标: 确定项目对研究人员和学术机构的价值程度。

度量指标	问题
学术开源项目的影响	学者或者学术团队创建的开源项目对大学教师的连任、职位任期和晋升过程有什么样的影响?

学术开源项目的影响

问题：学者或者学术团队创建的开源项目对大学教师的连任、职位任期和晋升过程有什么样的影响?

描述

学者需要在任职期间和晋升案例中展示其学术成果的举证。创建一个开源项目可能是一个有影响力的贡献，这个指标有助于证明这一点。该指标适用于新项目，并且这些新项目的创建是作为学术工作的一部分和以开源方式发布。该指标与对现有开源项目的开源贡献无关。

目标

目标是通过提出关键的开源影响指标来支持 RPT（任命、任职、晋升），例如：了解围绕创建的开源项目的社区范围了解开源项目的成长、成熟或衰退识别哪些项目依赖于您的项目识别引用您项目的期刊文章

实现

关于如何衡量学术开源项目影响的一些想法：

- 通过在《开源软件杂志》上发表来衡量影响
- 下游依赖的软件项目的数量
- 类似于 H 指数 (H-Index) 的东西（目前不存在）
- CiteAs API 为开源软件提供标准化的引用
- 下载次数
- 贡献者数量
- 软件/库引用数
- 星数 (GitHub)
- 引用软件或项目的已发表文章数量
- 下载引用软件或项目的预印本
- 更新规律
- 代码行
- 社区贡献的数量，不是来自研究团队
- 下游依赖

过滤条件

可视化

提供度量的工具

- GrimoireLab
- Augur
- CiteAS

- 开源软件杂志 (JOSS)
- GitHub Citation
- arXiv.org code
- 开放科学中心 OSF（开放科学框架）
- ACM Artifact Review and Badging

数据收集策略

参考资料

- GitHub 软件引用指南
- arXiv.org code
- 研究故事
- 替代指标
- 相关指标: 项目受欢迎程度
- 引用作为
- Zhao, R., & Wei, M. (2017). Impact evaluation of open source software: An Altmetrics perspective. Scientometrics, 110(2), 1017–1033. https://doi.org/10.1007/s11192-016-2204-y
- Moral-Muñoz, J. A., Herrera-Viedma, E., Santisteban-Espejo, A., & Cobo, M. J. (2020). Software tools for conducting bibliometric analysis in science: An up-to-date review. El Profesional de La Información, 29(1). https://doi.org/10.3145/epi.2020.ene.03
- Searles, A., Doran, C., Attia, J., Knight, D., Wiggers, J., Deeming, S., Mattes, J., Webb, B., Hannan, S., Ling, R., Edmunds, K., Reeves, P., & Nilsson, M. (2016). An approach to measuring and encouraging research translation and research impact. Health Research Policy and Systems, 14(1), 60. https://doi.org/10.1186/s12961-016-0131-2

贡献者

- Stephen Jacobs
- Vinod Ahuja
- Elizabeth Barron
- Matt Germonprez
- Kevin Lumbard
- Georg Link
- Sean P Goggins
- Johan Linaker

关注领域 - 公共价值

目标: 确定项目 (包括下游项目) 是否对社区用户或者贡献者有价值。

度量指标	问题
项目热度	开源项目有多受欢迎?
项目发展速度	如何衡量组织的发展速度?
社会聆听	根据定性情感，如何衡量社区互动的价值，并准确判定社区的 “声誉 “?

项目热度

问题: 开源项目有多受欢迎?

描述

项目受欢迎的程度可以通过项目周围可见的活跃度来衡量，受欢迎度有一个正向反馈循环，热度更高的项目获得更多的关注，吸引更多的开发者或者用户，并能看到热度持续增加，循环往复。

项目受欢迎度可以作为理解项目价值的关键指标，因为开源项目缺乏项目使用信息或者销售信息。

目的

为了赚取工资并最大限度的增加未来的就业机会，工作者可能有兴趣了解哪些项目正在增长。同样，从组织的角度来看，了解哪些项目使用率高有助于了解哪些项目可能值得投资。项目流行度指标可用于确定项目的发展轨迹。

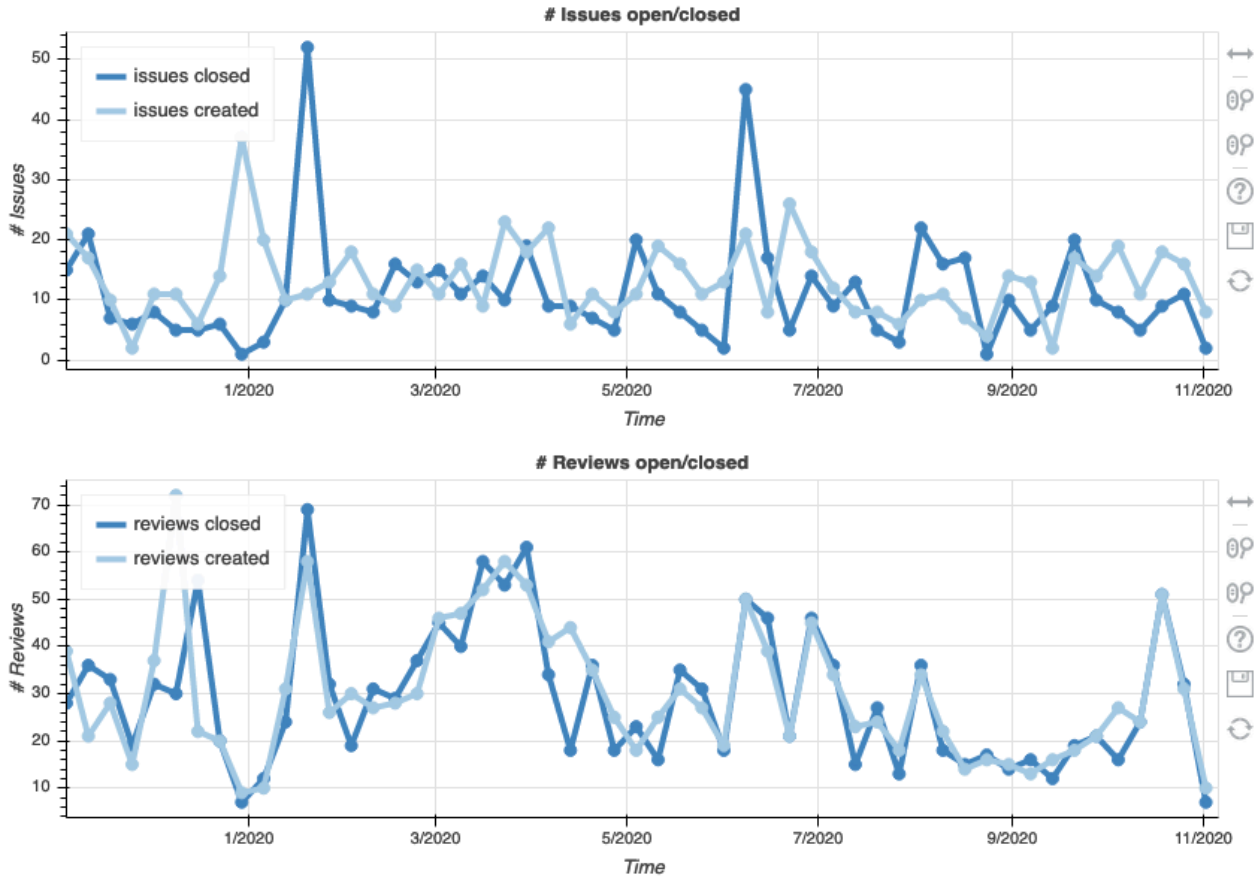
实现

项目受欢迎成都指标通常会随着时间的推移而发生变化。在根据以下数量衡量项目受欢迎程度时需要考虑大量的示例:

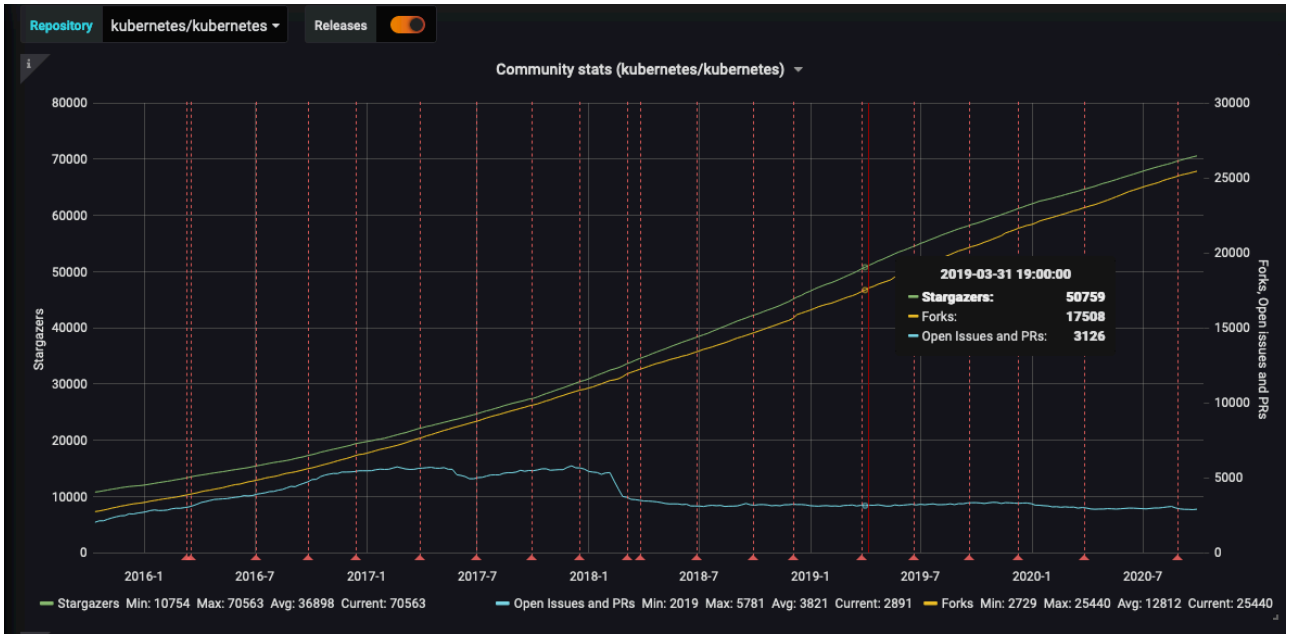
1. 社交媒体
2. 分支 (Forks)
3. 变更请求
4. 新议题
5. Stars, badges, likes
6. 新的贡献者
7. 组织多元化
8. 项目技能的职位发布
9. 项目内外的交流
10. 项目克隆数量
11. 追随者
12. 下游依赖
13. 参加项目活动的人

可视化

来自 Cauldron (GrimoireLab) 的议题和代码评审（变更请求）可视化:



来自 DevStats 的 Kubernetes 项目的流行度统计数据:



提供指标的工具

- Augur
- GrimoireLab
- Cauldron

参考

- 受欢迎的开源项目
- 是否得到维护?
- 开源项目趋势
- Kubernetes 工资

项目发展速度

问题：如何衡量组织的发展速度？

描述

项目的发展速度是指议题数量、代码提交数量、更改请求数量和贡献者个数，作为 “创新” 的指标。

目标

给开源项目办公室 (OSPO) 经理提供一种通过比较项目组合去衡量项目发展速度的方法。

OSPO 经理可以使用以下项目发展速度指标来：

- 比较开源项目与内部项目的发展速度
- 比较项目组合中的项目发展速度
- 找出哪些项目外部贡献者的发展超出了内部贡献者（在筛选内部贡献者与外部贡献者时）
- 找出值得投入的有前途的领域
- 找出未来几年可能成功的领域

参见示例

实现

基本指标包括：

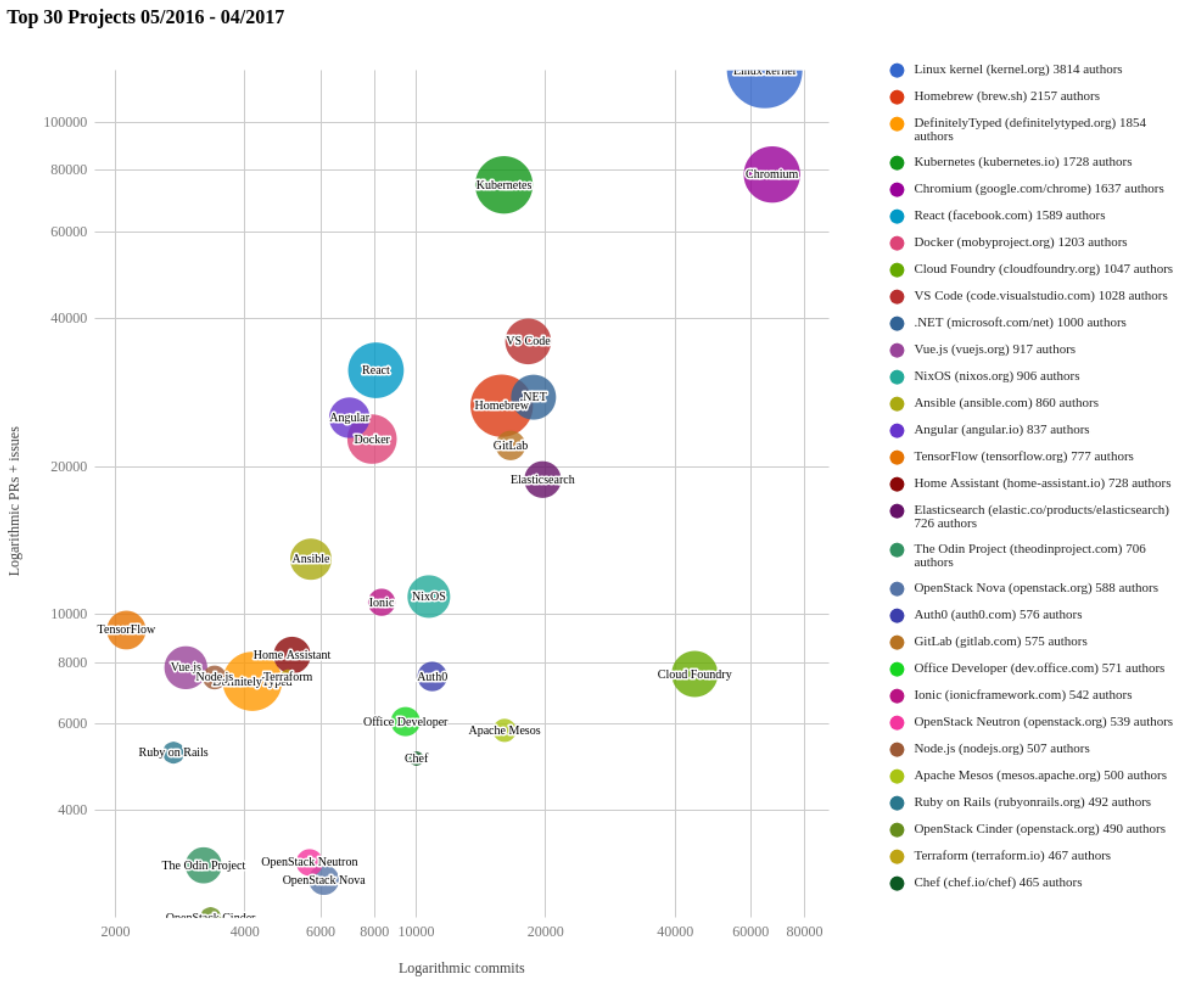
- 关闭的问题个数
- 评论个数
- 修改代码的人数
- 代码提交人数

筛选条件

- 内部与外部贡献者
- 项目来源（例如，内部仓库、开源仓库和竞争对手的开源仓库）
- 时间

可视化效果

- X 轴：代码提交的总次数的对数
- Y 轴：提交问题总个数和评论总个数之和的对数。
- 点大小：代码提交人数
- 点为某个项目



来自 CNCF

提供指标的工具

- CNCF - https://github.com/cncf/velocity

参考资料

- 开源创新能在企业中发挥作用吗?
- 高绩效文化的开放式创新
- 数字企业的开源
- 发展最快的开源项目

社会聆听

问题：根据定性情感，如何衡量社区互动的价值，并准确判定社区的 “声誉 “?

注意：这一指标综合了几种可以从跟踪数据中得出的其他指标，以及几种过程导向的指标。内嵌脚注注解了计划在以后澄清的领域和以后解决的问题。

描述

社会聆听将多渠道社群聆听实践与一组重要分类相结合。这些策略经过结合，可以形成系统的社区分析并提供社区战略信息，实现可衡量的业务价值。1

理论与起源

社交货币或社交资本是一种社会科学理论。它广泛研究了人类互动如何在社区中建立关系和信任。社交货币指标系统通过衡量社区信任度、透明度、实用性、一致性和荣誉来表示社区声誉。

人际关系是社区的社会结构。这可见于 Levinger 关系模型和社会渗透理论。社区成员的个人和群体认同感在互动中不断增强。成员通过自我表露行为建立共同的价值观，积累信任感，鼓励合作并互利互惠。这些互动建立了一种更强的并且可衡量的联系感。衡量这些特征的尺度被称为社交货币。2

结果

社交货币指标系统是从社区互动梳理 “消防水带 (fire hose)” 定性数据的方法。这种方法的中心前提是社区成员互动会对社区产生影响。社交货币指标系统会持续测量这些互动的情感 3。它说明了社区成员和负责人之间的声誉和信任程度。4

目标

分析社区互动中的定性评论。得出社区的情感概况。通过各项指标直观显示社区的现状和过去。从持续测量中使用领先指标主动制定社区策略。让社区成员相信他们的想法和意见能够得到重视。

实现

社会聆听需要收集社区评论（通信踪迹）、定义规范并对通信记录进行持续审查。5

如下 “工具” 一节所述，设置您所选择的数据收集平台。确保至少 4 个维度和 3 个通信渠道。设置完成后，采用以下方法收集、分析和解释结果：

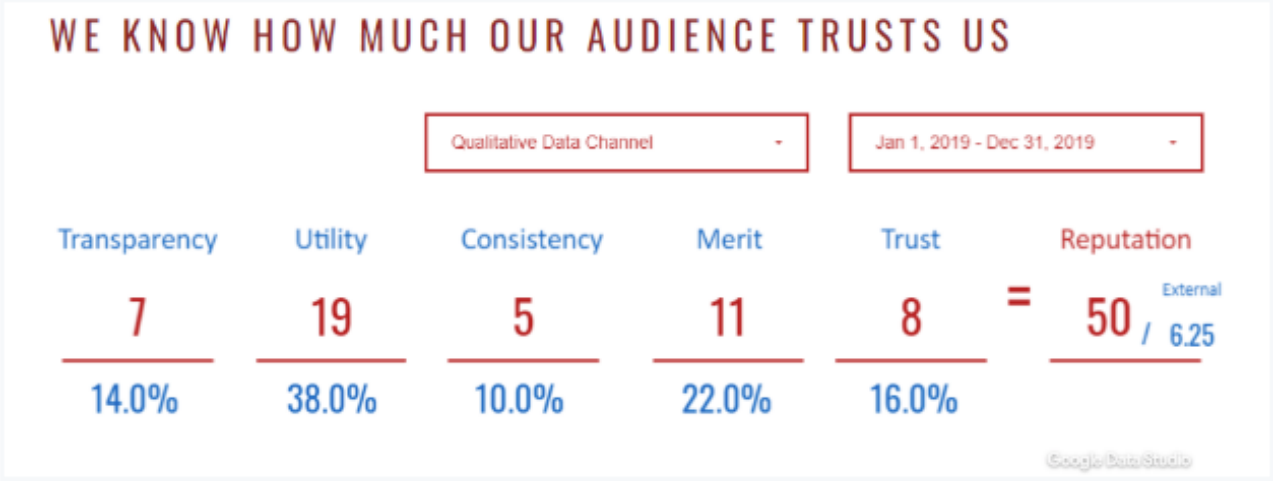
1. **收集通信踪迹** -- 找出您的社区在其上进行交流的在线平台。设置从主平台到社会聆听工具的数据漏斗。系统的关键数据是用户生成的内容。
2. **标准化通信踪迹的评估方式** -- 使用一套规范定义重要概念，作为焦点社区的 “跟踪关键词” 或 “类别”。在不同人阅读和标记社区情感时，术语统一规范能够确保分析的一致性。该规范必须定期修订和增补结构。5
3. **分析通信踪迹** -- 在社会聆听工具中，通过规范术语标记数据分析社区情感。如果是由团队进行标记，建议所有人定期集合讨论趋势，并确保标记使用的一致性。如果是由人工智能算法进行标记，则需要人类团队对 AI 进行必要的监督和再训练。5
4. **共享和呈现聚合分析** -- 在仪表板中呈现一定时间内规范术语的定量统计。定性分析结果在这里生成易于观察的趋势仪表板。与团队成员共享分析。6
5. **基准、设定目标与预测未来增长** -- 利用足够数据形成基准后，理清您的社区所处的位置。它有哪些优点和缺点？可以采取哪些措施让社区更健康、更稳固？然后形成目标明确的社区倡议并执行这些项目，对下周的社交货币指标产生影响。6
6. **重复流程** -- 在定期评估会议上讨论数据集或收集方法的不足之处。提出在未来弥补这些不足的方法。将解决方案纳入系统并继续推进。在趋势中寻求真理，在范式中汲取力量。7

筛选条件

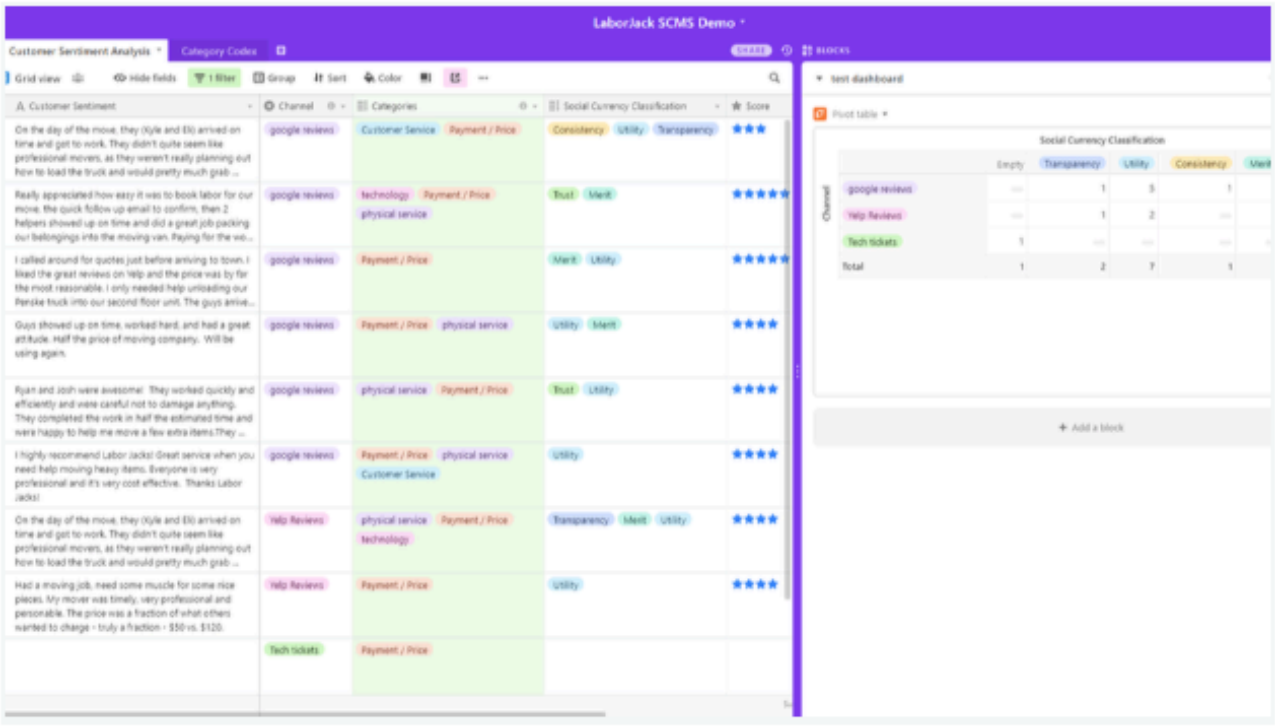
1. **渠道**：按数据收集处排序。
2. **标记**：根据评论中使用的情感识别规范标记显示数据。
3. **时间**：显示数据随时间变化的趋势，并拉取特定的数据集。
4. **最具影响力评论**：按标志排序和筛选，标志可放置于数据中以突出特定的数据点并说明其重要性。
5. **AI 与人类标记**：根据标记是由程序施加还是由人类施加来进行筛选。
6. ** 加权货币：** 根据任何一个单独选择的标准，对某些评论的 “重要性” 进行加权。得出的加权视图仅为信息基于权重的重新排序。

可视化效果

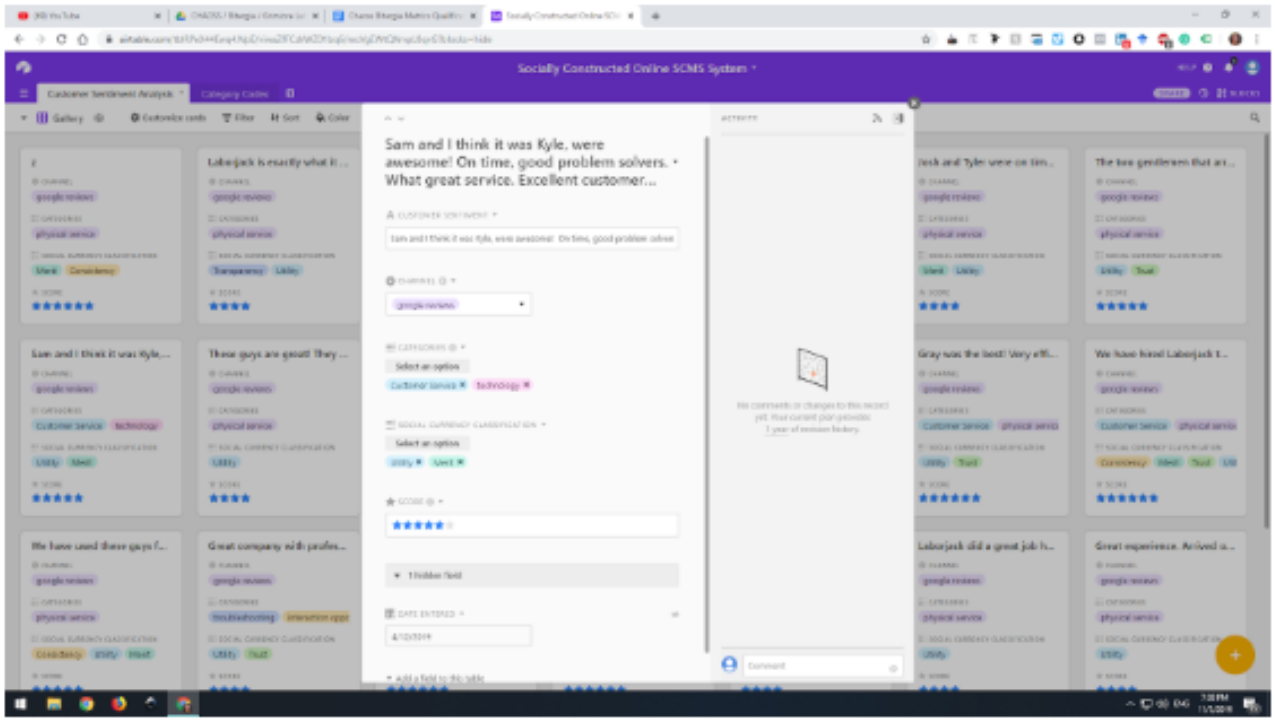
仪表板呈现聚合指标：



** 示例社会聆听工具：** 左侧为原始社区评论，标记添加于右侧紧邻的列中。右侧数据透视表以数字显示标记与其他标记组合出现的频率。



** 扩展评论视图：** 从字段中移除 “定量”，并提供阅读不同评论的最佳方式。



提供指标的工具

任何 MySQL、smart-sheet、excel 或类似 airtable 的 excel 数据表格程序都可以用于实现指标。这些数据应足够简化以与其他数据接口交互，确保数据迁移简单、直接并可自动进行（如 Google Data Studio）。这要求用于实现社会聆听的系统必须支持 CSV 和其他电子表格文件，我们强烈建议使用开源程序进行实现。

完成后，使用以下数据点创建数据集：8

Data Points	Description
Date of entry	Date data was imported to Social Listening tool
Date of comment	Date comment was made on original platform
Comment Text	Qualitative data brought in. Decide on how large you want these chunks ported. Some may port an entire email while others will be broken into one row per sentence. It should only have one “sentiment”
Data channel	Originating data channel the comment came from
Tags (created on codex document below)	Based on the unified codex of terms, decide what tags to track. There can be two kinds of tags. On the one hand, tags can be based on “themes” or recurring sentiment that people voice (e.g., gamer gate, flamewar, or thank you notes). On the other hand, tags based on “categories” can describe different aspects of a community that members comment on (e.g., events, release, or governance).
Social Currency Metric	The social currency being awarded or demerited in the system. This will directly affect numbers.
Weighted Score	Once you’ve decided what your “weight” will be, you can assign a system of -3 to +3 to provide a weighted view of human-tagged metrics (AI will not assign a weight for several reasons). This enables the “most impactful comment” filter.

为术语统一规范创建第二张表，对术语进行定义。如下所示：8

Category Term	Definition	When to use	When not to use
[Custom Tags - themes and categories]			
[Community specific jargon]			
Social Currency Dimensions:			
TRANSPARENCY	Do people recognize and feel a connection to your community?	When they have the "words" to pinpoint why they feel you are authentic or personable.	This is not about good customer service, or doing well. That is utility. This is about whether they understand who you are as a business and show they are onboard with it.
UTILITY	Is your community doing something useful or is it contributing value?	Provide parameters that exclude when the term is used so that people know when the category tag should not be implemented.	This is not about good customer service, or doing well. That is utility. This is about whether they understand who you are as a business and show they are onboard with it.
CONSISTENCY	Do you have a history of being reliable and dependable?	When they suggest they have used your brand, or interacted with you several times	If they've only provided their comment to suggest you were useful once, use utility instead.
MERIT	Does your community merit respect and attention for your accomplishments?	When the social currency garnered from customers seems it will continue for a while, and will impact other people's opinions.	When they suggest they will use you again in the future use trust instead as that is a personal trust in the brand. Merit is external.
TRUST	Can people trust that your community will continue to provide value and grow in the future?	When they suggest they trust you well enough to continue conversations with you in the future	When there is not substantial enough evidence to suggest they will continue to work with and trust you as a loyal customer or community member.
INTERNAL REPUTATION [9]	Do people believe these things strongly enough to warrant conversation or action?		
EXTERNAL REPUTATION [9]	What amount of your reputation in your community is transferable to strangers outside of your community (cold audiences)?		

该规范由利益相关方特定社区定期填写，构成数据分析的基础。这是最重要的部分。没有这一部分，定性数据的主观性就不符合普遍规则：9

> “A 概念仅在 C 限制下适用于 B 人群。”

数据收集策略

社区成员的评论可从跟踪数据中获得。理想情况下，社会聆听将评论文本自动导入标记工具。跟踪数据可以从社区成员编写评论的协作平台收集，这些平台包括票证系统、代码审查、电子邮件列表、即时消息、社交媒体和论坛。

法律和监管考量

销毁点：*xx* 月后，详细数据将被销毁。根据 GDPR 规定，定量计算最长可保存 250 周。超过 250 周的数据将成为无法处理但可以查看的归档数据。用户可以协商主要统计。

参考资料

- 在 airtable 上的示例实现
- 在 Data Studio 中的示例实现（报告）
- 在 Data Studio 中的示例实现（数据源）
- Google 表格中的示例实现
- 实现文档（始于第 33 页）

带注解的脚注

1 CHAOSS 指标一直以来用于建立标准定义，这些定义可在项目之间良好应用以支持比较。该指标可能会在未来演变为一个项目。

2 从此描述中得出什么指标？可能包括：1. 社区信任，2. 透明度，3. 实用性，4. 一致性，以及 5. 荣誉

3 情感分析表明，指标 (6) 很可能是 “通信情感”，定义可能需要引用常见情感分析工具，有时被称为 “词袋”。

4 衡量如何向社区成员灌输信任，使他们的思想和意见得到重视，这种可能的指标 (7) 将定义一项进程，也许无法通过跟踪数据衡量。

5 开源软件的任意规范中都有相当一部分是各项目之间共有的，每个项目都可能有一套特定兴趣属于规范的一个子集。在某些情况下，其主要兴趣可能不会出现在既定的规范组成部分中。通常，规范和 CHAOSS 项目本身一样作为共享元数据开源，确保跨开源社区的共享理解。

6 这描述了标准规范的演变及其通过 CHAOSS 工作组和项目进程的要素，特征如上一个脚注所述。这可能为进程指标 (8)。

7 候选进程导向指标 (9)。

8 使用开源规范编码的数据示例经过发展，将成为通过社会聆听推进开源软件的核心组成部分。实现将需要这些示例，作为 CHAOSS 项目的开源资产提供，并将以共享数据返回价值。

9 内部和外部声誉可能是社会聆听产生的指标 (10) 和 (11)。

关注领域 - 个人价值

目标: 识别一个项目对我作为个人用户或贡献者是否有价值

度量指标	问题
组织项目技能需求	有多少组织在使用这个项目？如果我精通这个项目，是否可以雇用我?
就业机会	有多少招聘信息要求项目技术与技能?

组织项目技能需求

问题：有多少组织在使用这个项目？如果我精通这个项目，是否可以雇用我？

描述

各组织通过用例和依赖关系来参与开源项目。该指标旨在确定与开源项目相关的下游技能需求。该指标着眼于将项目作为 IT 基础设施的一部分进行部署的组织、其他具有声明依赖关系的开源项目，以及通过社交媒体、会议提及、博客文章和类似活动对项目的引用。

目标

作为一名开发者，我希望把自己的技术和时间投入到有可能让我在未来获得一份体面薪水的项目中。人们可以使用项目软件的下游组织影响指标来了解组织使用的项目，由此寻求可能需要 IT 支持服务的工作机会。

实现

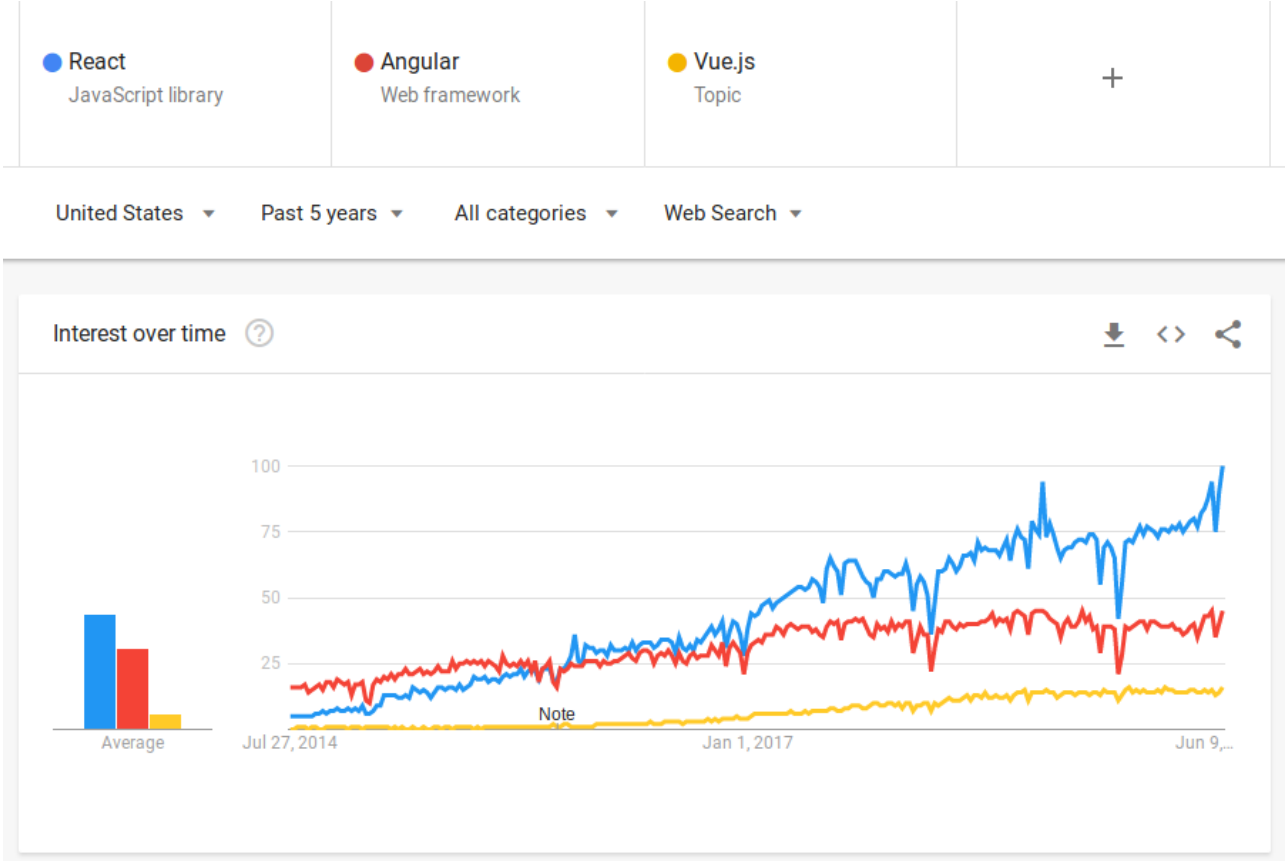
基本指标包括：

- 为项目创建问题的组织数量
- 为项目创建拉取请求的组织数量
- 发布项目相关博客或推文的组织数量
- 在公开招聘信息中提到项目的组织数量
- 派代表参加项目相关会议的组织数量
- 依赖某项目的其他项目数
- 项目相关书籍的数量
- 项目的 Google 搜索趋势

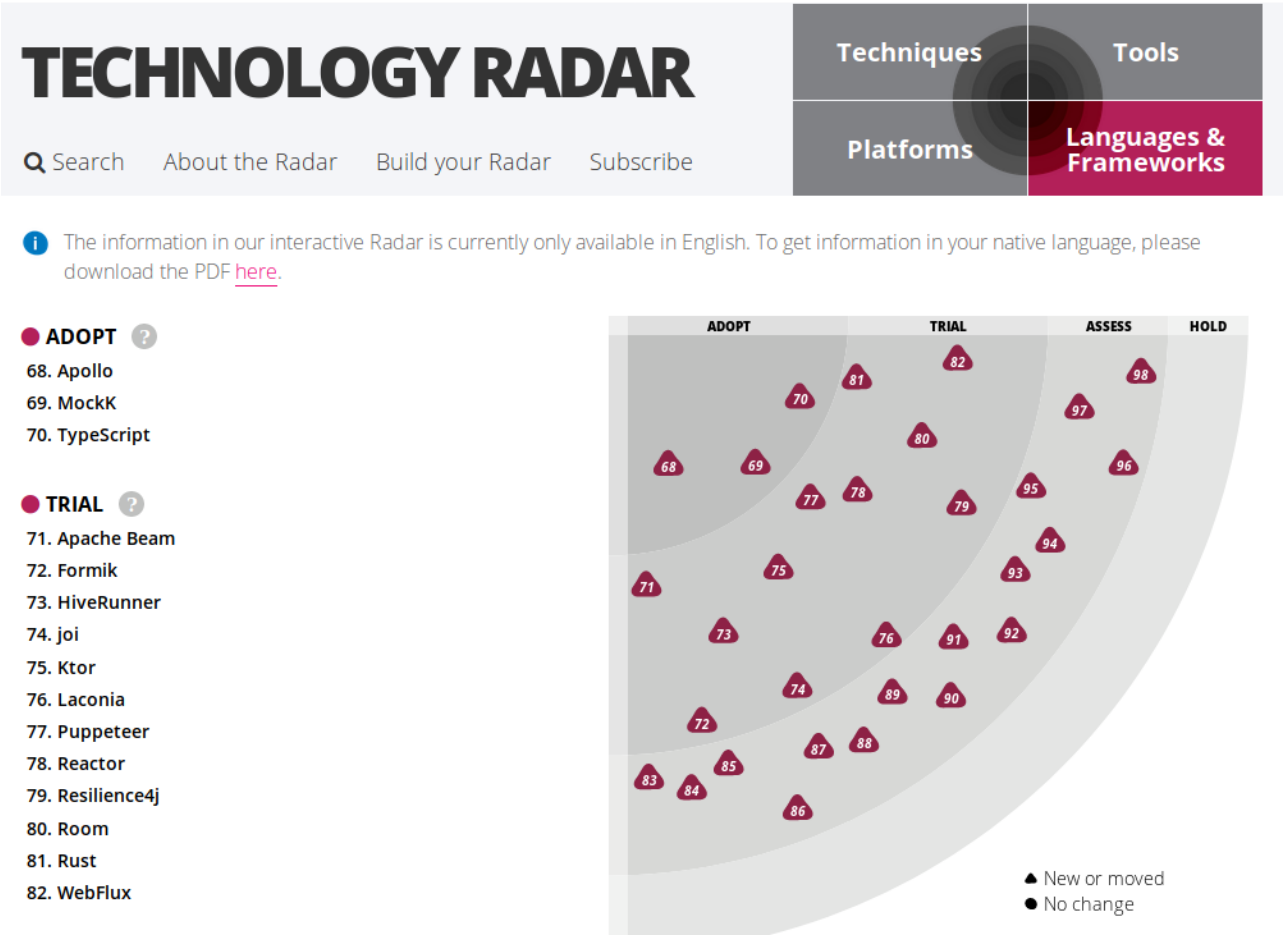
可视化效果

以下可视化效果展示了依赖于特定项目的下游项目的数量。虽然这种可视化效果未捕捉项目软件下游组织影响指标的整体，但它提供了一部分的可视化效果。

其他可视化效果可能包括 Google 搜索趋势（React vs. Angular vs. Vue.js）。



ThoughtWorks 发布了名为'Tech Radar' 的系列，展现技术的普及程度。



Tech Radar 让您能够对项目进行深入研究，了解评估随时间的变化。

React.js

NOV 2016

ADOPT

In the avalanche of front-end JavaScript frameworks, **React.js** stands out due to its design around a reactive data flow. Allowing only one-way data binding greatly simplifies the rendering logic and avoids many of the issues that commonly plague applications written with other frameworks. We're seeing the benefits of React.js on a growing number of projects, large and small, while at the same time we continue to be concerned about the state and the future of other popular frameworks like AngularJS. This has led to React.js becoming our default choice for JavaScript frameworks.

NOT ON THE CURRENT EDITION

This blip is not on the current edition of the radar. If it was on one of the last few editions it is likely that it is still relevant. If the blip is older it might no longer be relevant and our assessment might be different today. Unfortunately, we simply don't have the bandwidth to continuously review blips from previous editions of the radar

Understand more »

APR 2016

ADOPT

NOV 2015

TRIAL

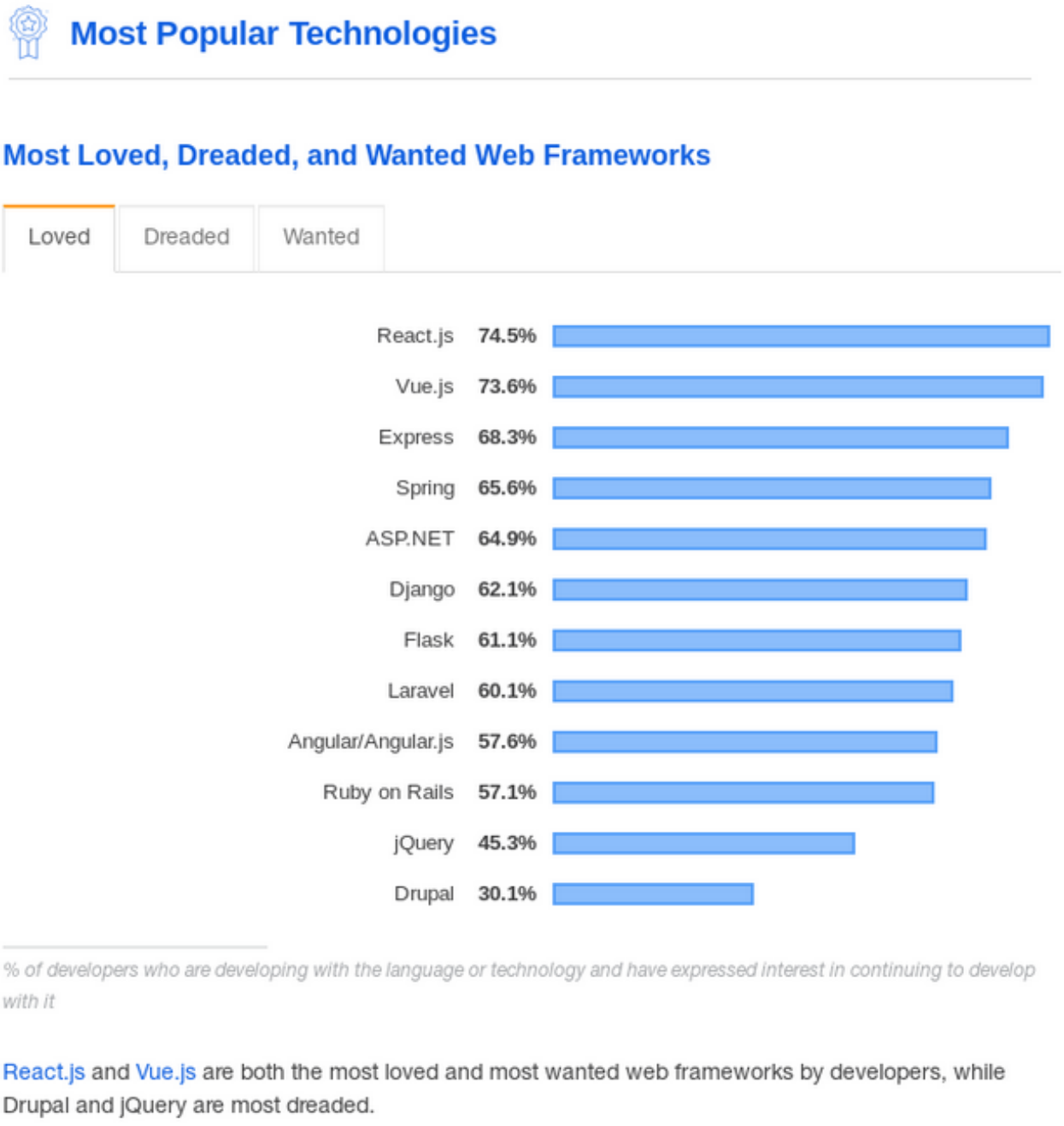
One benefit of the ongoing avalanche of front-end JavaScript frameworks is that occasionally a new idea crops up that makes us think. **React.js** is a UI/view framework in which JavaScript functions generate HTML in a reactive data flow. It differs significantly from frameworks like AngularJS in that it only allows one-way data bindings, greatly simplifying the rendering logic. We have seen several smaller projects achieve success with React.js, and developers are drawn to its clean, composable approach to componentization.

MAY 2015

TRIAL

One benefit to the ongoing avalanche of front-end JavaScript frameworks is that occasionally, a new idea crops up that makes us think. **React.js** is a UI/View framework in which JavaScript functions generate HTML in a reactive data flow. We have seen several smaller projects achieve success with React.js and developers are drawn to its clean, composeable approach to componentization.

StackOverview 发布了年度开发者调查



提供指标的工具

- Google 趋势 - 显示一段时间内的搜索兴趣
- ThoughtWorks TechRadar - 技术咨询公司的项目评估
- StackOverflow 开发者调查 - 年度项目排名
- Augur；示例可用于多个仓库：
  - Rails
  - Zephyr
  - CloudStack

参考资料

- 开源赞助
- 财政赞助和开源
- 大型企业开源赞助
- Google 趋势 API
- 衡量开源软件的影响
- ThoughtWorks Tech Radar
- Stack Overflow 开发者调查

就业机会

问题：有多少招聘信息要求项目技术与技能?

描述

开源贡献者赚取生活费的一种常见方式是受雇于公司或成为自雇或自由开发者。特定项目中的技能可能会改善求职者获得工作的前景。与特定开源项目中所学技能相关的最明显需求指标是该项目或其技术纳入招聘信息中的时间。

目标

该指标可以让贡献者了解在特定开源项目中所学技能被公司重视的程度。

实现

要在求职平台（如 LinkedIn、Indeed 或 Dice）上获得这一指标，前往职位搜索并输入开源项目的名称。返回的招聘信息数量即为指标。通过求职平台的 API 定期收集指标并存储结果，即可查看趋势。

筛选条件

- 招聘信息时长；信息会过时，填写后可能不会被移除

可视化效果

可以查看以下内容扩展指标：

- 返回的职位的薪资范围
- 返回的职位的资历水平
- 现场或非现场职位的可用性
- 工作地点
- 地理位置

参考资料

- LinkedIn 职位搜索 API： https://developer.linkedin.com/docs/v1/jobs/job-search-api#
- Indeed 职位搜索 API： https://opensource.indeedeng.io/api-documentation/docs/job-search/
- Dice.com 职位搜索 API： http://www.dice.com/external/content/documentation/api.html
- Monster 职位搜索 API： https://partner.monster.com/job-search
- Ziprecruiter API（需要合作）： https://www.ziprecruiter.com/zipsearch

_ 注意：_ 此指标仅限于单个项目，但参与开源也可由其他原因获益。这一指标可以超越单一项目，改为使用编程语言、进程、开源经验或框架等相关技能作为职位的搜索参数。

关注领域 - 人力投资

目标: 从组织的角度看项目是否具有经济价值。

度量指标	问题
组织影响力	一个组织对一个社区有多大影响?
人力投资	组织投入人力对社区所做的贡献（例如：代码提交，议题和更改请求）花费的成本是多少

组织影响力

问题: 一个组织对一个社区有多大影响?

描述

组织影响力是衡量一个组织对开源社区的影响力的一个尺度。组织通常是较大的企业公司，可能会影响项目的方向，表明在社区中有某种程度的控制。这相当于一个组织对其参与并拥有既得利益的开源项目的开发轨迹的影响程度。

目标

组织能在开源社区中产生影响。对组织影响力的衡量本身就具有启发性，也可以作为一个显示组织影响力随时间增长的机会。这个指标可以帮助组织中的开源倡导者证明继续为参与提供资金和支持是合理的。该指标还可以帮助开源维护人员跟踪和测量组织的影响，以增加其项目的合法性，并监控各个组织的控制水平。组织影响力也可以作为一个信号，表明新成员为开源项目做出贡献可能是容易的也可能是困难的。

实现

在衡量组织影响力时要考虑的一些具体例子包括:

- 有多少组织成员为一个项目做出贡献 - 贡献者
- 组织成员的级别和类型 贡献者类型
- 以与社区活动水平成比例的高比率提交变更请求的组织。例如，其他贡献组织的一个标准差或更少。有关 组织多样性
- 技术指导委员会的组织成员
- 理事会组织成员
- 项目维护者角色中的组织成员
- 赞助开源项目的组织

可视化

提供度量的工具

- Augur
- GrimoireLab

数据收集策略

参考

贡献者

- Sean Goggins
- Matt Germonprez
- Vinod Ahuja

- Kevin Lumbard
- Lawrence Hecht
- Matt Snell
- Dhruv Sachdev
- Elizabeth Barron
- Matt Broberg
- Stephen Jacobs

人力投资

问题：组织投入人力对社区所做的贡献（例如：代码提交，议题和更改请求）花费的成本是多少

描述

开源项目通常由组织的人力投入来支撑。该指标跟踪组织对单个项目的经济投入（体现在人力成本）。

目标

随着组织参与度对开源项目变得越来越重要，组织必须清楚了解其人力投资。该指标的目的是为从事开源项目的组织提高人力成本的透明度。该指标给开源项目办公室（OSPO）经理提供了一种通过项目投资组合比较人力成本的方法。比如：人力投资指标能用在确定投资的优先顺序或者确定投资回报。例如：

- 以人力投资评估 OSPO 事务的优先级和证明预算合理性
- 以人力投资解释产品、项目管理事项的优先级
- 以人力投资论证继续投资 OSPO 的价值
- 以人力投资反应和比较开源贡献与内部工作的人力成本
- 以人力投资比较项目组合的项目效益

实现

基础指标包括：

- 贡献数量
- 按贡献者类型（内部/外部）划分的贡献数量
- 按贡献类型（如代码提交，议题和更改请求）划分的贡献数量

参数包括：

- 每小时劳动率
- 创建贡献的平均劳动时间（按照贡献类型分类）

人力投资 = 每一种贡献类型的总和（贡献数量 * 创造贡献的平均工时 * 平均每小时劳动率）

筛选条件

- 内部与外部贡献者
- 问题标签
- 项目来源（如内部、开源仓库、竞争对手的开源仓库）

可视化效果

```
1 IssueID, Severity, Title          , Status, Contributor, Tag
2 34234   , High    , Add CSV Graphic, Open  , andyl      , metrics
3 23421   , Med     , Fix typos      , Closed, mattg      , metrics
4 56743   , High    , Reword section , Open  , georg      , augur
5 85879   , Low     , Add CNCF PNG   , Open  , seang      , metrics
6 34183   , High    , Remove button  , Closed, vinod      , implementation
7 76790   , Low     , Use large font , Open  , kevin      , metrics
8 57432   , Med     , Sync with web  , Closed, carol      , implementation
```

我们的第一个参数化指标的可视化效果依赖于可以用 Augur 导出的 CSV。电子表格用于指标参数和计算公式。未来的实现可能会在 webapp 中直接添加参数操作的功能。

参考资料

- 启动开源项目办公室
- 创办开源项目办公室
- 企业开源

Release Notes

CHAOSS metrics are released continuously. The regular release is when we update the version number, update the full release notes, and make a big announcement. These releases occur one to two times a year and may correspond with the dates of a CHAOSScon event. Prior to regular release, continuous released metrics go through a comment period of at least 30 days.

Release 2021-10 Notes:

- PDF of released CHAOSS Metrics (v.2021-10)
- **Common WG**
  - Focus Area Name Revisions:
    - People (was ”Who”)
    - Contribution (was ”What”)
    - Time (was ”When”)
    - Place (was ”Where”)
  - New metrics include:
    - Bot Activity
    - Clones
    - Collaboration Platform Activity
    - Event Locations
    - Language Distribution
  - Metric Revision:
    - Technical Forks
- **Diversity & Inclusion WG**
  - New metrics include:
    - Documentation Discoverability
    - Inclusive Experience at Event
    - Psychological Safety
  - Name Change/Revision:
    - Attendee Demographics and Speaker Demographics merged into Event Demographics
- **Evolution WG**
  - New metrics include:
    - Contribution Attribution
- **Risk WG**
  - New Focus Area:
    - Dependency Risk Assessment
  - New metrics include:
    - Libyears
    - Upstream Code Dependencies
- **Value WG**
  - New Focus Area:
    - Academic Value
  - New metrics include:
    - Academic Open Source Project impact
    - Organizational Influence

CHAOSS Contributors

Aastha Bist, Abhinav Bajpai, Ahmed Zerouali, Akshara P, Akshita Gupta, Amanda Brindle, Anita Ihuman, Alberto Martín, Alberto Pérez García-Plaza, Alexander Serebrenik, Alexandre Courouble, Alolita Sharma, Alvaro del Castillo, Ahmed Zerouali, Amanda Casari, Amy Marrich, Ana Jimenez Santamaria, Andre Klapper, Andrea Gallo, Andy Grunwald, Andy Leak, Aniruddha Karajgi, Anita Sarma, Ankit Lohani, Ankur Sonawane, Anna Buhman, Armstrong Foundjem, Atharva Sharma, Ben Lloyd Pearson, Benjamin Copeland, Beth Hancock, Bingwen Ma, Boris Baldassari, Bram Adams, Brian Proffitt, Camilo Velazquez Rodriguez, Carol Chen, Carter Landis, Chris Clark, Christian Cmehil-Warn, Clement Li, Damien Legay, Dani Gellis, Daniel German, Daniel Izquierdo Cortazar, David A. Wheeler, David Moreno, David Pose, Dawn Foster, Derek Howard, Don Marti, Drashti, Duane O'Brien, Dylan Marcy, Eleni Constantinou, Elizabeth Barron, Emily Brown, Emma Irwin, Eriol Fox, Fil Maj, Gabe Heim, Georg J.P. Link, Gil Yehuda, Harish Pillay, Harshal Mittal, Henri Yandell, Henrik Mitsch, Igor Steinmacher, Ildiko Vancsa, Jacob Green, Jaice Singer Du Mars, Jaskirat Singh, Jason Clark, Javier Luis Cánovas Izquierdo, Jeff McAffer, Jeremiah Foster, Jessica Wilkerson, Jesus M. Gonzalez-Barahona, Jilayne Lovejoy, Jocelyn Matthews, Johan Linåker, John Coghlan, John Mertic, Jon Lawrence, Jonathan Lipps, Jono Bacon, Jordi Cabot, Jose Manrique Lopez de la Fuente, Joshua Hickman, Joshua R. Simmons, Josianne Marsan, Justin W. Flory, Kate Stewart, Katie Schueths, Keanu Nichols, Kevin Lumbard, King Gao, Kristof Van Tomme, Lars, Laura Dabbish, Laura Gaetano, Lawrence Hecht, Leslie Hawthorne, Luis Cañas-Díaz, Luis Villa, Lukasz Gryglicki, Mariam Guizani, Mark Matyas, Martin Coulombe, Matthew Broberg, Matt Germonprez, Matt Snell, Michael Downey, Miguel Ángel Fernández, Mike Wu, Neil Chue Hong, Neofytos Kolokotronis, Nick Vidal, Nicole Huesman, Nishchith K Shetty, Nithya Ruff, Nuritzi Sanchez, Parth Sharma, Patrick Masson, Peter Monks, Pranjal Aswani, Pratik Mishra, Prodromos Polychroniadis, Quan Zhou, Ray Paik, Remy DeCausemaker, Ria Gupta, Richard Littauer, Ritik Malik, Robert Lincoln Truesdale III, Robert Sanchez, RoyceCAI Rupa Dachere, Ruth Ikegah, Saicharan Reddy, Saloni Garg, Saleh Abdel Motaal, Samantha Lee, Samantha Venia Logan, Samson Goddy, Santiago Dueñas, Sarit Adhikari, Sarvesh Mehta, Sarah Conway, Sean P. Goggins, Shane Curcuru, Sharan Foga, Shaun McCance, Shreyas, Silona Bonewald, Sophia Vargas, Sri Ramkrishna, Stefano Zacchiroli, Stefka Dimitrova, Stephen Jacobs, Tharun Ravuri, Thom DeCarlo, Tianyi Zhou, Tobie Langel, Saleh Abdel Motaal, Tom Mens, UTpH, Valerio Cosentino, Venu Vardhan Reddy Tekula, Vicky Janicki, Victor Coisne, Vinod Ahuja, Vipul Gupta, Will Norris, Xavier Bol, Xiaoya Xia, Yash Prakash, Yehui Wang, zhongjun2, Zibby Keaton

Are you eligible to be on this list? You are if you helped in any capacity, for example: Filed an issue. Created a Pull Request. Gave feedback on our work. Please open an issue or post on the mailing list if we've missed anyone.

CHAOSS Governing Board Members

- Amy Marrich, Red Hat
- Andrea Gallo, Linaro
- Armstrong Foundjem, MCIS Laboratory at Queen’s University
- Daniel Izquierdo, Bitergia
- Dawn Foster, VMware
- Don Marti, CafeMedia
- Georg Link, Bitergia
- Ildikó Vancsa, OpenStack
- Kate Stewart, Linux Foundation
- Matt Germonprez, University of Nebraska at Omaha
- Nicole Huesman, Intel
- Ray Paik, GitLab
- Sean Goggins, University of Missouri
- Wayne Beaton, Eclipse Foundation

LICENSE

MIT License