Metrics

Release 2021-04

https://chaoss.community/metrics

SAMPLE REPORT

CHAOSS Contributors include:
Aastha Bist, Abhinav Bajpai, Ahmed Zerouali, Akshara P, Akshita Gupta, Amanda Brindle, Alberto Martín, Alberto Pérez García-Plaza, Alexander Serebrenik, Alexandre Courouble, Alolita Sharma, Alvaro del Castillo, Ahmed Zerouali, 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, Bingwen Ma, Boris Baldassari, Bram Adams, Brian Proffitt, Camilo Velazquez Rodriguez, Carol Chen, Carter Landis, Chris Clark, Christian Cmehil-Warn, Damien Legay, Dani Gellis, Daniel German, Daniel Izquierdo Cortazar, David A. Wheeler, David Moreno, David Pose, Dawn Foster, Derek Howard, Don Marti, Drashti, Dylan Marcy, Eleni Constantinou, Elizabeth Barron, 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, Jason Clark, Javier Luis Cánovas Izquierdo, Jeff McAffer, Jeremiah Foster, Jessica Wilkerson, Jesus M. Gonzalez-Barahona, Jilayne Lovejoy, Jocelyn Matthews, Johan Linåker, 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, 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, Robert Lincoln Truesdale III, Robert Sanchez, Rupa Dachere, 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, 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, Zibby Keaton

The CHAOSS Governing Board at time of release:

Contents

Technical Fork

Question: What are a number of technical forks of an open source project on code development platforms?

Description

A technical fork is a distributed version control copy of a project. The number of technical forks indicates the number of copies of a project on the same code development platform.

Objectives

The objective of the Technical Fork metric is to ascertain how many copies of a project exist on a code development platform. Analysis of technical forks may provide insight into forking intentions (different types of forks such as contributing, and non-contributing forks).

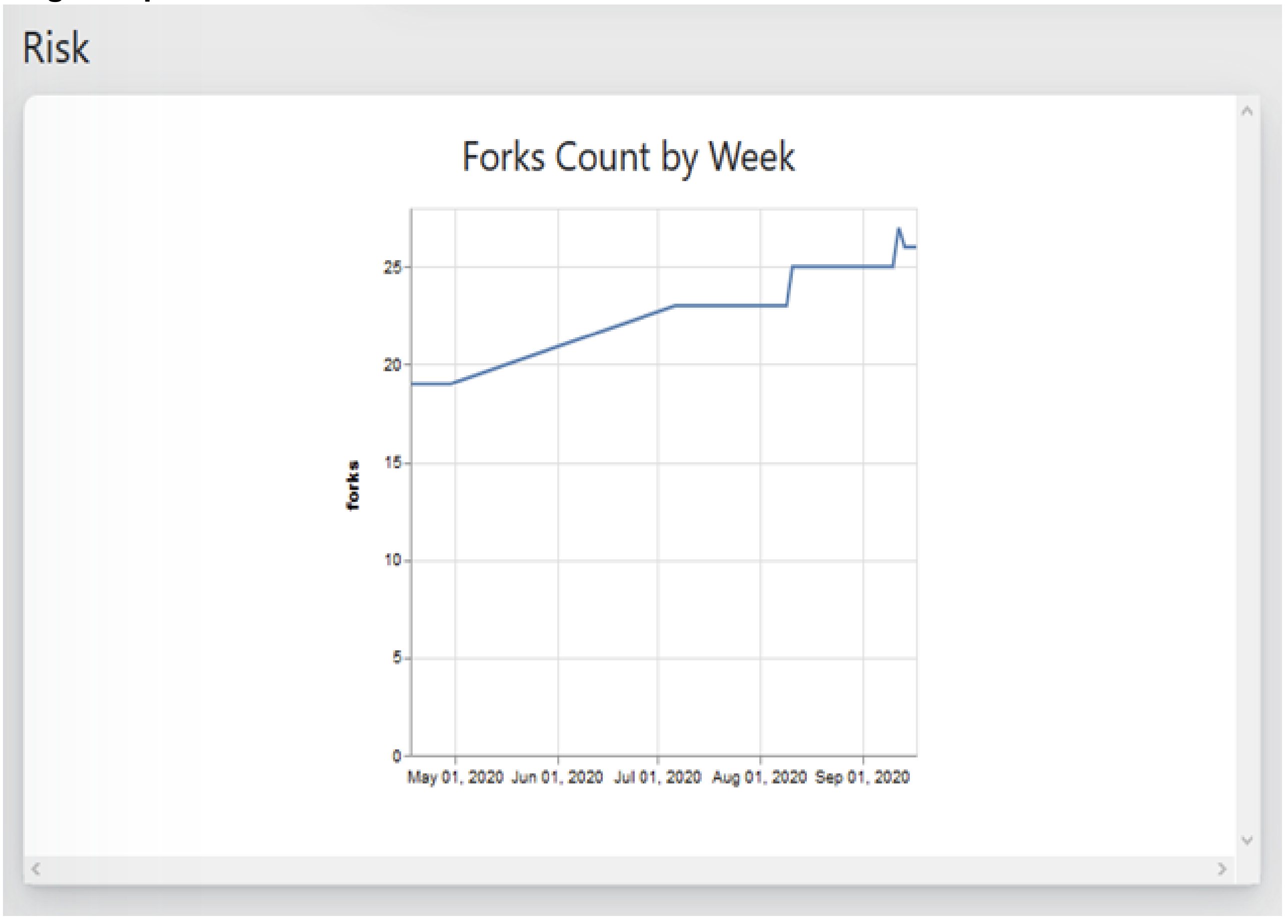
Implementation

Filters

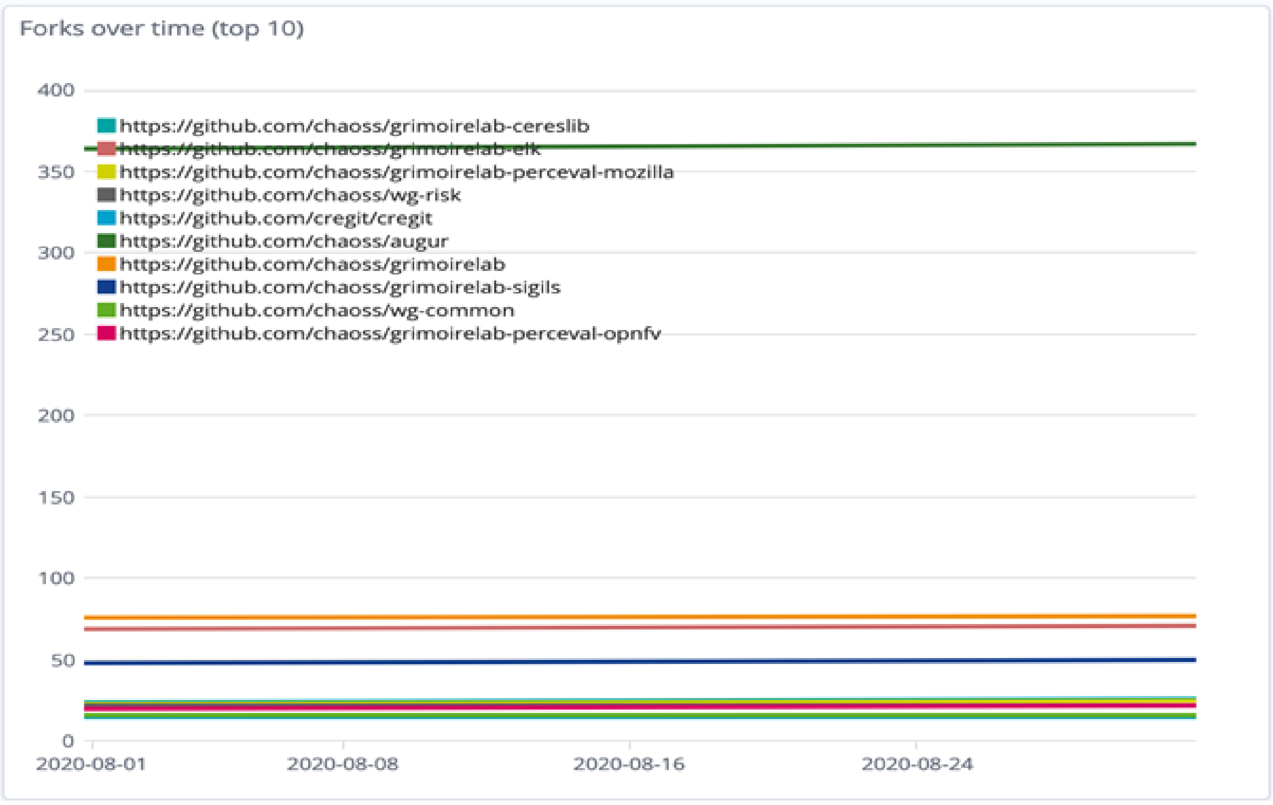
- Time Period (e.g., Weekly, Monthly, Annually)
- Ratio of contributing fork to total forks (A contributing fork is a fork that has opened a change request against the original repository.)
- Ratio of non-contributing fork to total forks (A non-contributing fork is a fork that has never opened a change request against the original repository.)

Visualizations

Augur Implementation



GrimoireLab Implementation



Tools Providing the Metric

- Augur
- GrimoireLab

Data Collection Strategies

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

References

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

Types of Contributions

Question: What types of contributions are being made?

Description

Multiple, varied contributions make an open source project healthy. Many projects have community members who do not write code but contribute in equally valuable ways by managing the community, triaging bugs, evangelizing the project, supporting users, or helping in other ways.

Objectives

A variety of contribution types can demonstrate that a project is mature and well-rounded with sufficient activity to support all aspects of the project, and enable paths to leadership that are supportive of a variety of contribution types and people with varying expertise beyond coding.

Implementation

How contributions are defined, quantified, tracked and made public is a challenging question. Answers may be unique to each project and the following suggestions are a starting point. As a general guideline, it is difficult to compare different contribution types with each other and they might better be recognized independently.

- The following list can help with identifying contribution types:
  - Writing Code
  - Reviewing Code
  - Bug Triaging
  - Quality Assurance and Testing
  - Security-Related Activities
  - Localization/L10N and Translation
  - Event Organization
  - Documentation Authorship
  - Community Building and Management
  - Teaching and Tutorial Building
  - Troubleshooting and Support
  - Creative Work and Design
  - User Interface, User Experience, and Accessibility
  - Social Media Management
  - User Support and Answering Questions
  - Writing Articles
  - Public Relations - Interviews with Technical Press
  - Speaking at Events
  - Marketing and Campaign Advocacy
  - Website Development
  - Legal Council
  - Financial Management

Data Collection Strategies

- **Interview or Survey:** Ask community members to recognize others for their contributions to recognize contribution types that have previously not been considered.
  - Who in the project would you like to recognize for their contributions? What did they contribute?
- **Observe project:** Identify and recognize leads of different parts of the project.
  - What leaders are listed on the project website or in a repository?
- **Capture Non-code Contributions:** Track contributions through a dedicated system, e.g., an issue tracker.
  - Logging can include custom information a project wants to know about non-code contributions to recognize efforts.
  - Proxy contributions through communication channel activity. For example, If quality assurance members (QA) have their own mailing list, then activity around QA contributions can be measured by proxy from mailing list activity.
- **Collect Trace Data:** Measure contributions through collaboration tool log data.
  - For example, code contributions can be counted from a source code repository, wiki contributions can be counted from a wiki edit history, and email messages can be counted from an email archive
- **Automate Classification:** Train an artificial intelligence (AI) bot to detect and classify contributions.
  - An AI bot can assist in categorizing contributions, for example, help requests vs. support provided, or feature request vs. bug reporting, especially if these are all done in the same issue tracker.

Other considerations:

- Especially with automated reports, allow community members to opt-out and not appear on the contribution reports.
- Acknowledge imperfect capture of contribution types and be explicit about what types of contributions are included.
- As a project evolves, methods for collecting types of contributions will need to adapt. For example, when an internationalization library is exchanged, project activity around localization conceivably produces different metrics before and after the change.
- Account for activity from bots when mining contribution types at large scale.

References

- 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

Activity Dates and Times

Question: What are the dates and timestamps of when contributor activities occur?

Description

Individuals engage in activities in open source projects at various times of the day. This metric is aimed at determining the dates and times of when individual activities were completed. The data can be used to probabilistically estimate where on earth contributions come from in cases where the time zone is not UTC.

Objectives

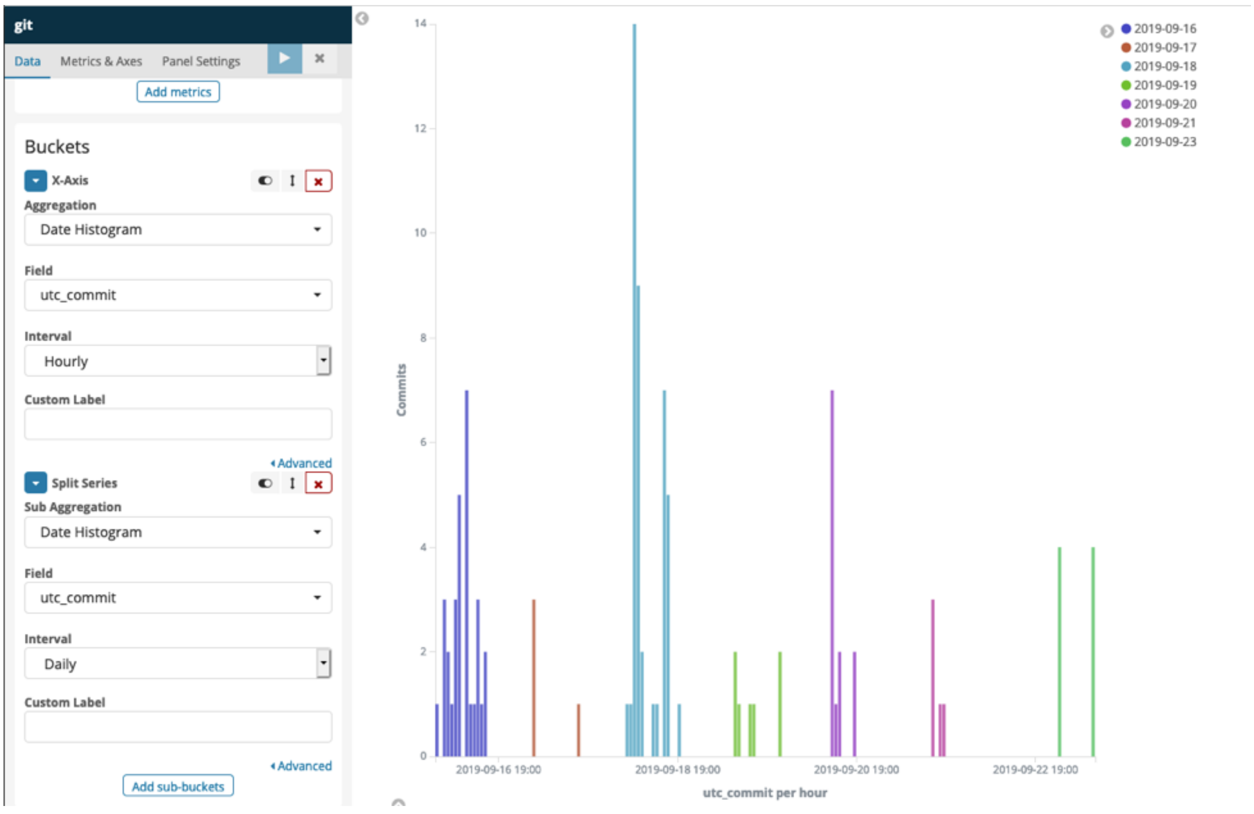
- Improve transparency for employers about when organizational employees are engaging with open source projects
- Improve transparency for open source project and community managers as to when activity is occurring

Implementation

Filters

- Individual by Organization
- Aggregation of time by UTC time
  - Can show what times across the globe contributions are made; when the project is most active.
- Aggregation of time by local time
  - Can show what times of day in their local times they contribute. Conclusions about the If contributions are more during working hours, or if contributions are more during evening hours.
- Repository ID
- Segment of a community, (e.g., GrimoireLab has more EU time zones activity and Augur more US time zones activity)

Visualizations



Buckets

X-Axis

Aggregation: Date Histogram

Field: utc_commit

Interval: Hourly

Custom Label:

Split Series

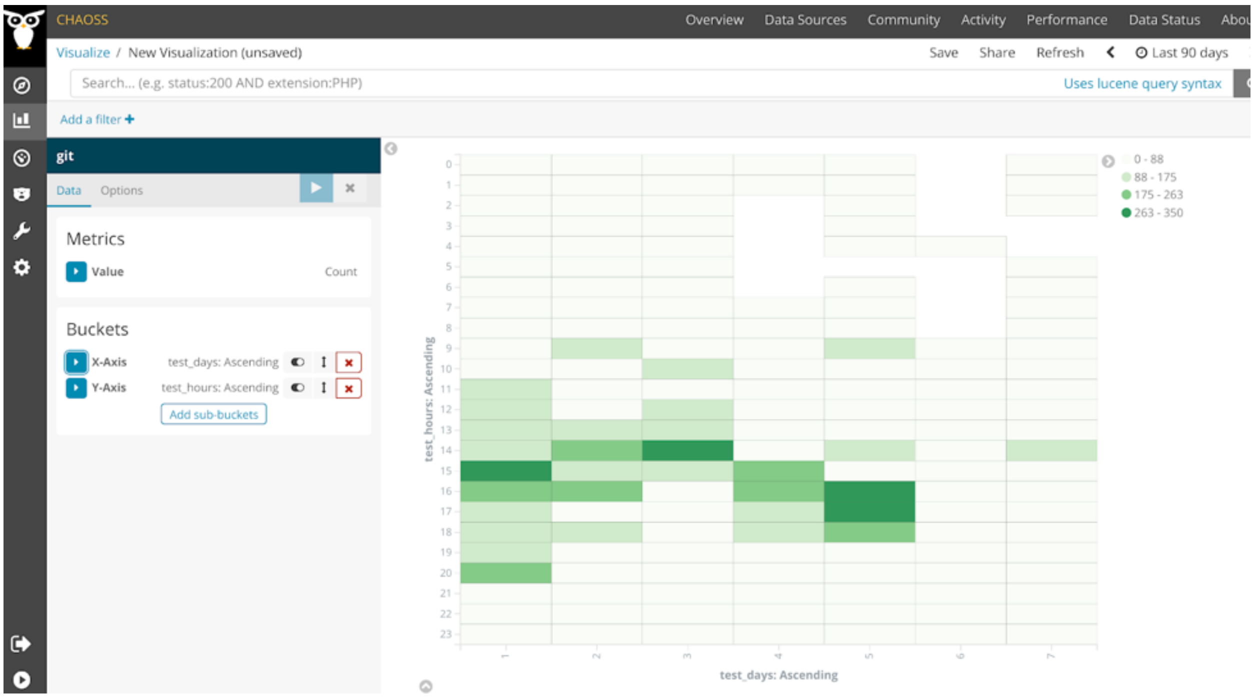
Sub Aggregation: Date Histogram

Field: utc_commit

Interval: Daily

Custom Label:

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



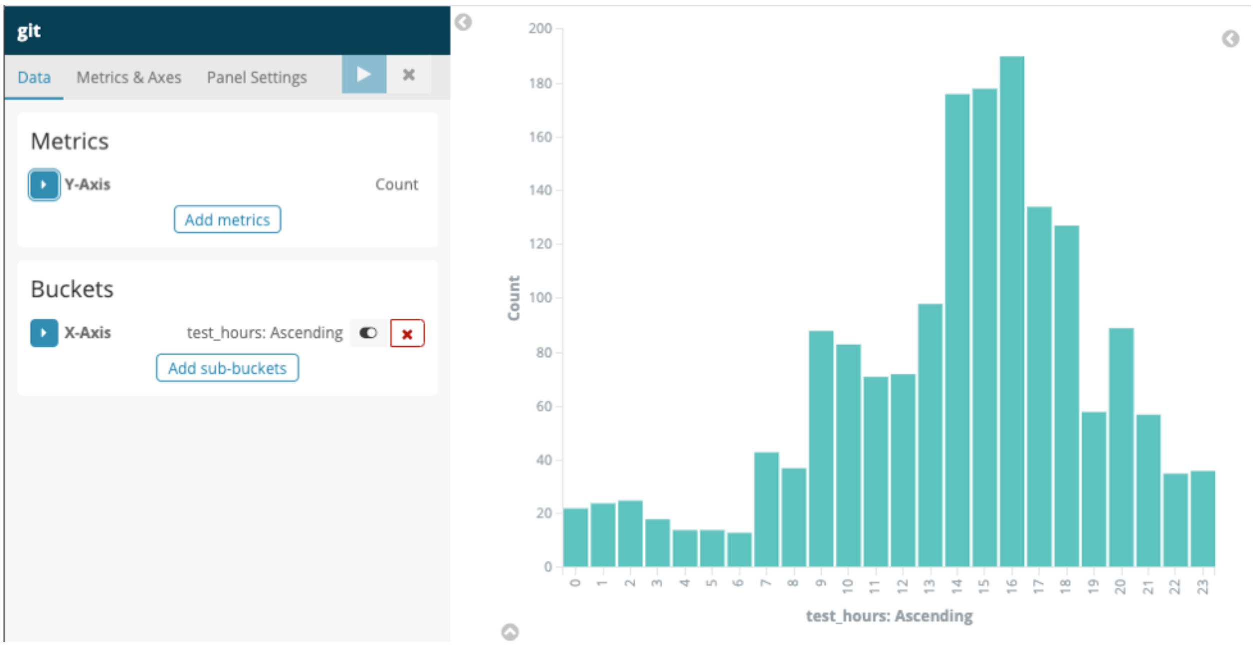
Metrics

Value: Count

Buckets

X-Axis test_days: Ascending

Y-Axis test_hours: Ascending



Tools Providing Metric

GrimoireLab

Augur Date/Timestamps

References

Coordinated Universal Time

Burstiness

Question: How are short timeframes of intense activity, followed by a corresponding return to a typical pattern of activity, observed in a project?

Description

There are a number of reasons that may prompt a sudden increase or decrease in the amount of activity within a repository. These increases and decreases appear both as a sudden change in activity against the average amount of activity. Burstiness is a way of understanding the cycle of activity in existing metrics, like issues, merge requests, mailing lists, commits, or comments. Examples of root causes for bursts in activity include:

- Release cycles
- Global pandemics
- Hackathon activities
- Mentorship programs
- Conferences, meetups, and other events where tools are presented
- Conventional and social media announcements and mentions
- Critical bugs as raising awareness and getting people's attention
- Community design meetings or brainstorming meetings to address a particular issue
- Community members show up from another community that is relying on your project (e.g., dependencies)

Objectives

- To identify impacts of root causes of a burst in activity
- To provide awareness when project activity unknowingly goes up
- To help capture the meaningfulness of increases or decreases in project activity
- To help the community and maintainers prepare for future bursts that follow a pattern
- To help measure the impact of influential external activities
- To differentiate skewed activity versus normal activity

Implementation

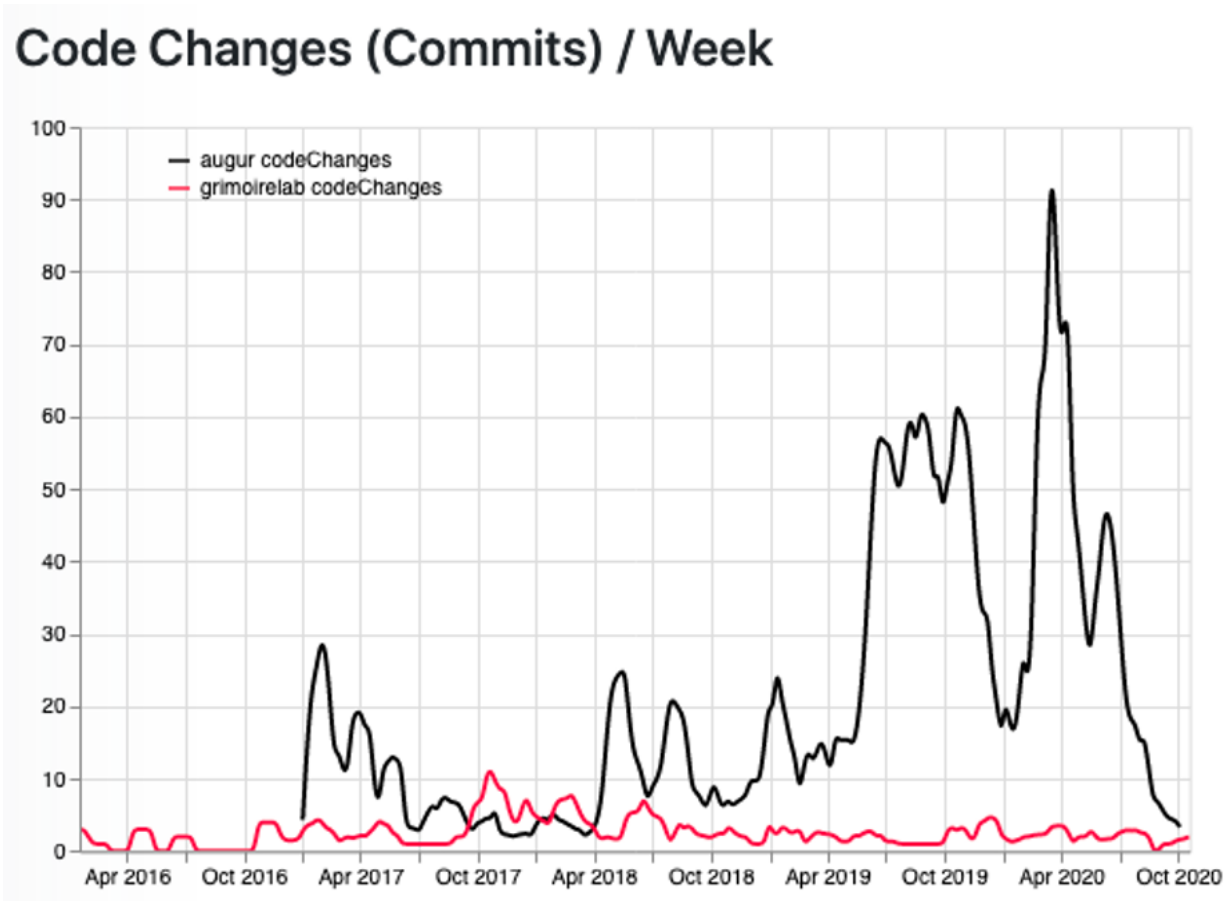
Filters

- Stars
- Forks
- Issues or bug reports
- Labels
- Downloads
- Release Tags
- Change Requests
- Mail List Traffic
- Documentation additions or revisions
- New Repositories
- Feature Requests
- Messaging Conversations
- Conventional and Social Media Activity

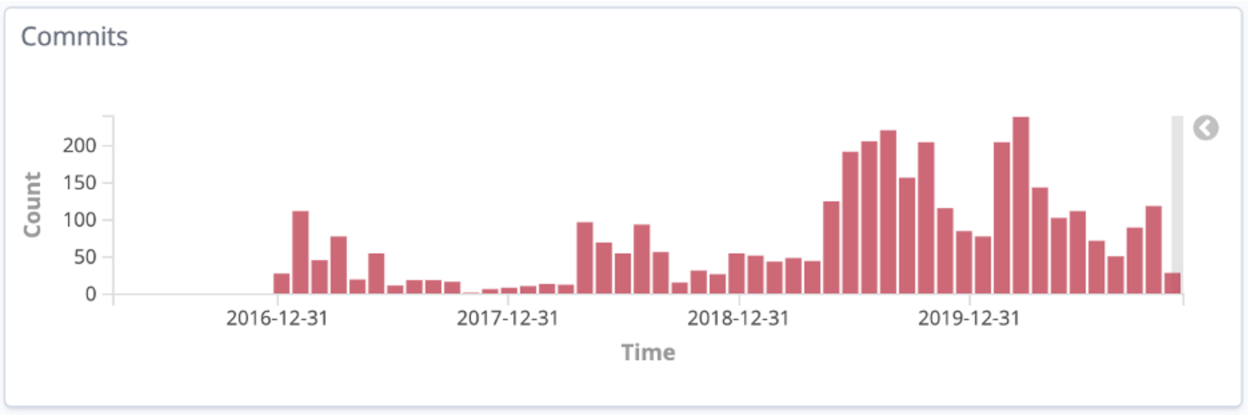
- Conference Attendance and Submissions

Visualizations

Augur:



GrimoireLab:



Tools Providing the Metric

- Grimoire Lab
- Augur

Data Collection Strategies

- Quantitative
    - Time box activities identifying deviations away from some norm
    - Outliers for certain thresholds, using statistics like Bollinger Bands to measure stability or volatility: https://en.wikipedia.org/wiki/Bollinger_Bands
- Qualitative Interview Questions
    - Why do you contribute more during a period of time?
    - What do you believe to be the root cause for particular bursts?
    - What impact do different events (e.g., hackathons, mentorship program, or conferences) have on project activity?

References

This metric was inspired by the work of Goh and Barabasi (2008): https://arxiv.org/pdf/physics/0610233.pdf

Review Cycle Duration within a Change Request

Question: What is the duration of a review cycle within a single change request?

Description

A change request is based on one or more review cycles. Within a review cycle, one or more reviewers can provide feedback on a proposed contribution. The duration of a review cycle, or the time between each new iteration of the contribution, is the basis of this metric.

Objectives

This metric provides maintainers with insight on: Code review process decay, as there are more iterations and review cycle durations increase. Process bottlenecks resulting in long code review iterations. Abandoned or semi-abandoned processes in the review cycles, where either the maintainer or the submitter is slow in responding. Characteristics of reviews that have different cyclic pattern lengths.

Implementation

Review Cycle Duration is measured as the time length of one review cycle within a single change request. The duration can be calculated between: The moment when each review cycle begins, defined as the point in time when a change request is submitted or updated. The moment when each review cycle ends, either because the change request was updated and needs a new review or because it was accepted or rejected.

Filter

Average or Median Duration, optionally filtered or grouped by: Number of people involved in review Number of comments in review Edits made to a change request Project or program Organization making the change request Time the change request was submitted Developers who contributed to a change request Change request Number of review cycle on a change request (e.g., filter by first, second, … round)

Visualizations

Tools Providing the Metric

References

Example of data that could be used to develop the metric: https://gerrit.wikimedia.org/r/c/mediawiki/core/+/194071

Time to Close

Question: How much time passes between creating and closing an operation such as an issue, change request, or support ticket?

Description

The time to close is the total amount of time that passes between the creation and closing of an operation such as an issue, change request, or support ticket. The operation needs to have an open and closed state, as is often the case in code review processes, question and answer forums, and ticketing systems.

Related metric: Issue Resolution Duration

Objectives

1. Determining how responsive a community is can help efforts to be inclusive, attract, and retain new and existing contributors.

2. Identifying characteristics of operations that impact an operation closing quickly or slowly (e.g., finding best practices, areas of improvement, assess efficiency).

3. Identifying bias for timely responses to different community members.

4. Detecting a change in community activity (e.g., to indicate potential maintainer burnout, reduction in the diversity of contributions)

5. Understand how the time to close an issue or change request is related to merge success or failure.

Implementation

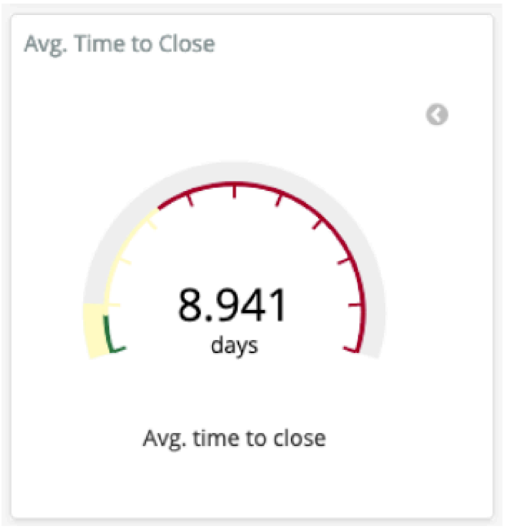
Filters

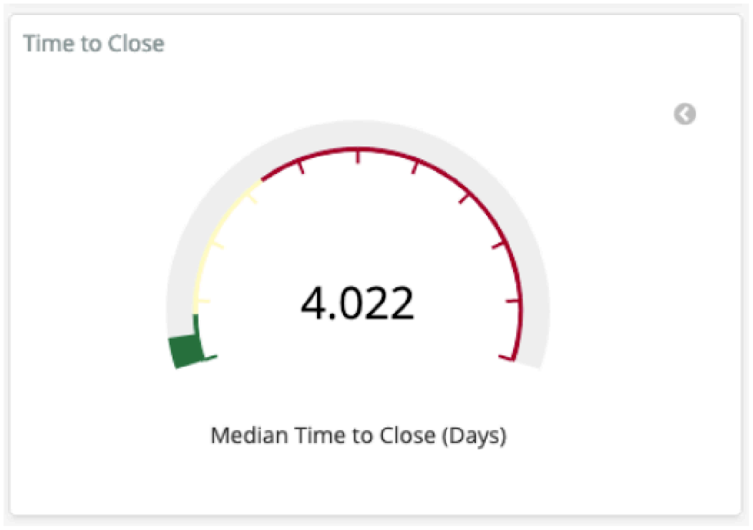
- Creator of operation (e.g., new contributor vs. maintainer)

- First closed, final closed

- Labels (e.g., bug vs. new feature)
- Change Request Merge Status (e.g. Time to Merge or Time to Close without Merge)

Visualizations





Tools Providing the Metric

Augur implementation:

- Issue Close Duration
- Issue Duration
- Issue Response Time

GrimoireLab implementation:

- Pull Requests Efficiency
- Issues Efficiency
- Efficiency:TimingOverview

Data Collection Strategies

The time to close metric may be contextual based on the project activity and objectives. For example, the time to close a bug report may provide different information than the time to close a new feature request. Data collection strategies should address different project objectives. Other variables that may influence these processes are:

- Issue Tracking Systems: the type of issue such as bug report, blueprint (OpenStack nomenclature), user story, feature request, epic, and others may influence how long this event takes to be closed. Other variables, such as the priority or severity may help to advance how quickly this event will be closed.

- Change Request Processes: this depends on the change request infrastructure, as Gerrit, GitHub or mailing lists (as in the Linux Kernel) and may differ depending on how complicated the process is. For example, newcomers or advanced and experienced developers will proceed in different ways and with more or less time required.

- Question and Answer Forum: this depends on the quality of the answer and the opinion of the person asking the question. A valid answer is marked, and the process is closed once the person questioning has successfully found a correct answer to their question.

References

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

Time to First Response

Question: How much time passes between when an activity requiring attention is created and the first response?

Description

The first response to an activity can sometimes be the most important response. The first response shows that a community is active and engages in conversations. A long time to respond to an activity can be a sign that a community is not responsive. A short time to respond to an activity can help to engage more members into further discussions and within the community.

Objectives

Identify cadence of first response across a variety of activities, including PRs, Issues, emails, IRC posts, etc. Time to first response is an important consideration for new and long-time contributors to a project along with overall project health.

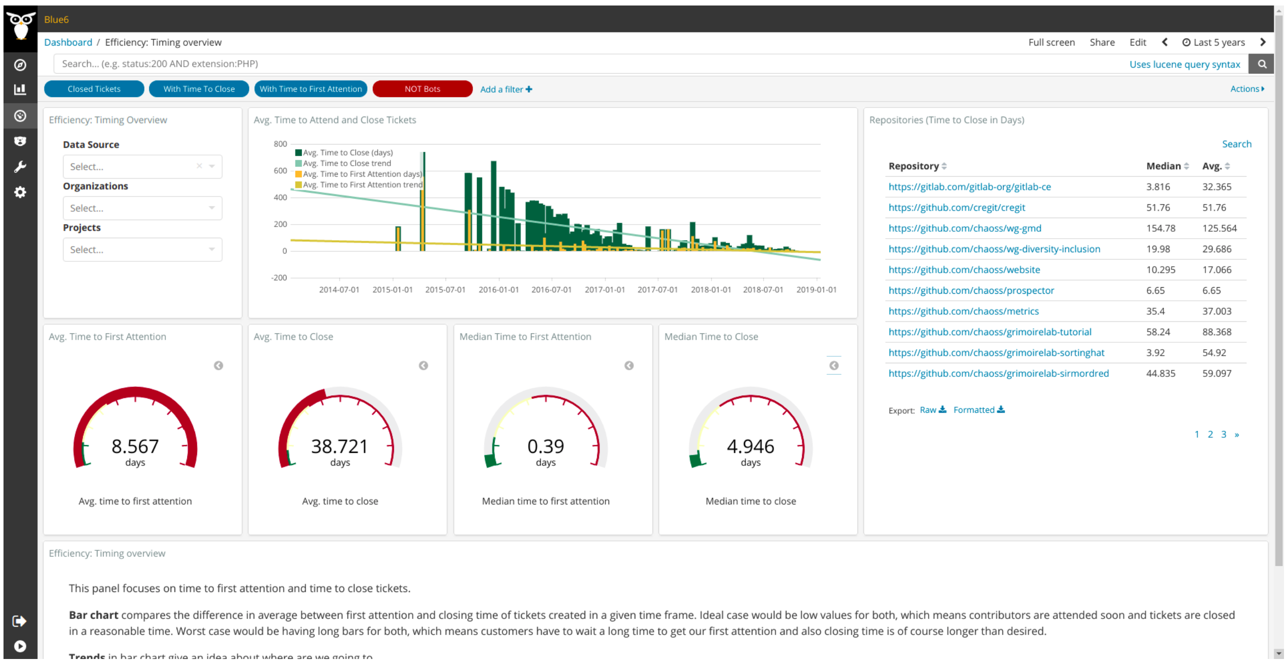
Implementation

Time to first response of an activity = time first response was posted to the activity - time the activity was created.

Filters

- Role of responder, e.g., only count maintainer responses
- Automated responses, e.g., only count replies from real people by filtering bots and other automated replies
- Type of Activity, e.g., issues (see metric Issue Response Time), emails, chat, change requests

Visualizations


Blue6
Dashboard / Efficiency: Timing overview
Full screen
Share
Edit
Last 5 years
Search... (e.g. status:200 AND extension:PHP)
Uses lucene query syntax
Closed Tickets
With Time To Close
With Time to First Attention
NOT Bots
Add a filter
Actions
Efficiency: Timing Overview
Data Source
Organizations
Projects
Select...
Avg. Time to Attend and Close Tickets
Avg. Time to Close (days)
Avg. Time to Close trend
Avg. Time to First Attention (days)
Avg. Time to First Attention trend

Repositories (Time to Close in Days)

Repository	Median	Avg.
https://gitlab.com/gitlab-org/gitlab-ce	3.816	32.365
https://github.com/cregit/cregit	51.76	51.76
https://github.com/chaoss/wg-gmd	154.78	125.564
https://github.com/chaoss/wg-diversity-inclusion	19.98	29.686
https://github.com/chaoss/website	10.295	17.066
https://github.com/chaoss/prospector	6.65	6.65
https://github.com/chaoss/metrics	35.4	37.003
https://github.com/chaoss/grimoirelab-tutorial	58.24	88.368
https://github.com/chaoss/grimoirelab-sortinghat	3.92	54.92
https://github.com/chaoss/grimoirelab-sirmordred	44.835	59.097

Export: Raw Formatted

1 2 3 »

Avg. Time to First Attention	Avg. Time to Close	Median Time to First Attention	Median Time to Close
8.567 days	38.721 days	0.39 days	4.946 days
Avg. time to first attention	Avg. time to close	Median time to first attention	Median time to close

Efficiency: Timing overview

This panel focuses on time to first attention and time to close tickets.

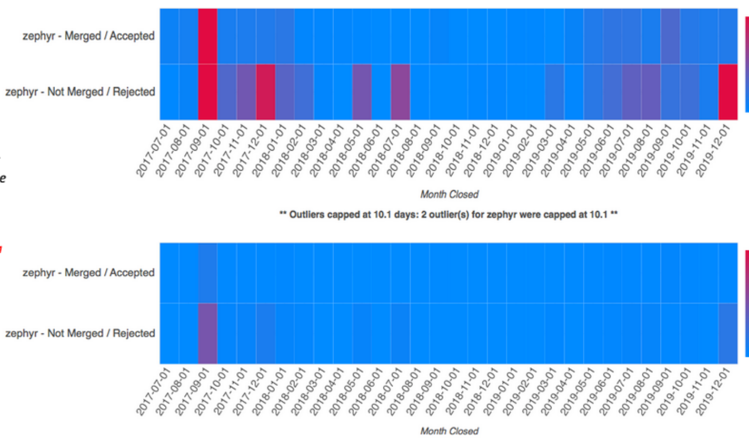
Bar chart compares the difference in average between first attention and closing time of tickets created in a given time frame. Ideal case would be low values for both, which means contributors are attended soon and tickets are closed in a reasonable time. Worst case would be having long bars for both, which means customers have to wait a long time to get our first attention and also closing time is of course longer than desired.

Trends in bar chart give an idea about where are we going to

Mean Days to First Response for Closed Pull Requests

Some Internal Slowing, But Outperforming Other Repositories

The heat maps illustrate variance in mean pull request duration to the first response for Zephyr over time. The left side compares Zephyr against itself, and the heat map on the right side is on a scale more aligned with the competition. Note especially that the internal Zephyr gradient has a max of 10 days, while the external comparisons have a max of 75 days.

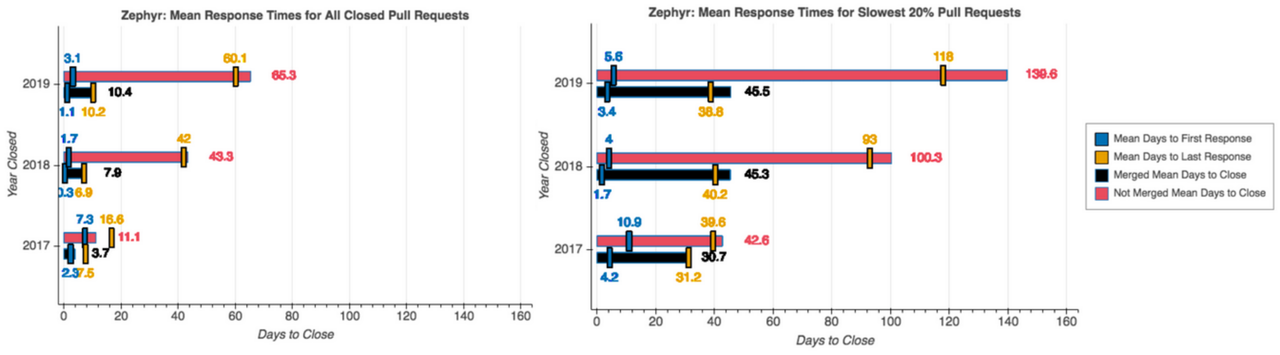


Internal Zephyr PR Performance (0 - 10 scale)

Zephyr PR Performance Compared to Others (0 - 75 scale)

Mean Response Times (Days) For Closed Pull Requests

Long Running Pull Requests Are Usually Rejected



The length of the black bar illustrates the total number of days that rejected and merged pull requests were open. The blue bar shows that, for the Zephyr project, we see the mean time to first response improving significantly for both rejected and merged pull requests from 2017 - 2019. The orange bar shows the last response or event associated with a pull request.

Tools Providing the Metric

- GrimoireLab Panel: Efficiency Timing Overview
- Kata Containers dashboard efficiency panel

References

Contributor Location

Question: What is the location of contributors?

Description

Geographical location from which contributors contribute, where they live, or where they work.

Objectives

To determine global locations of contributors in an effort to understand work practices and times zones. To identify where contributions do not come from in an effort to improve engagement in these areas.

Implementation

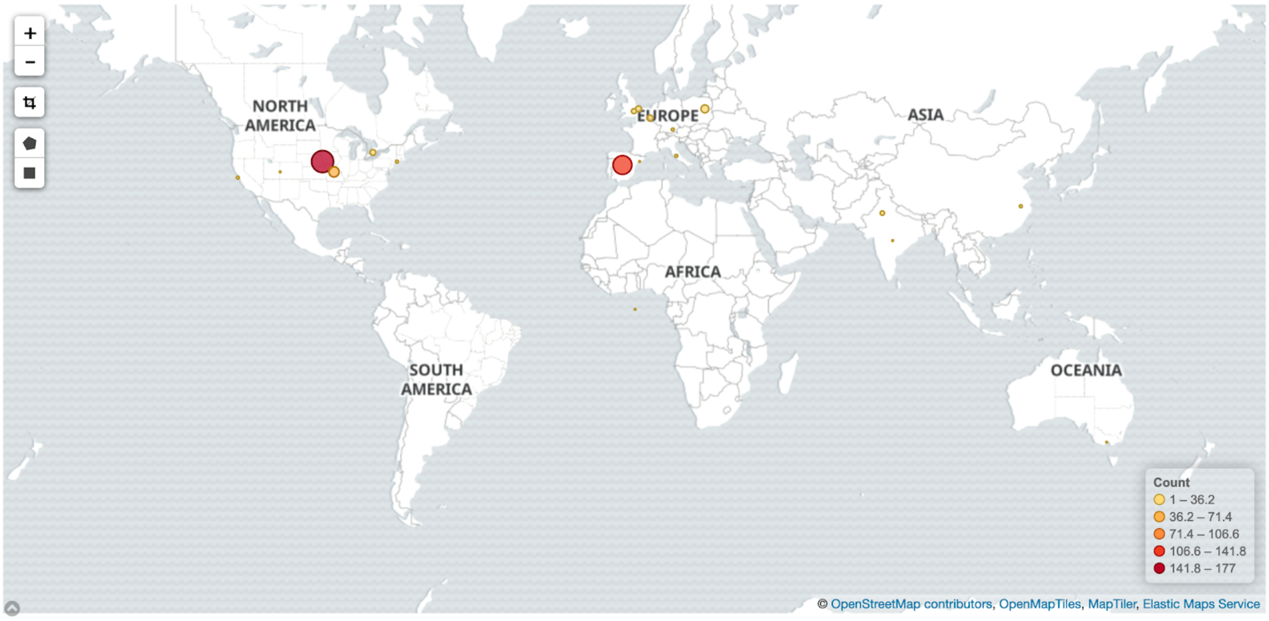
Filters

Filter contributions by:

- **Location.** Attempt to group locations in regions to have multiple levels of reporting. Location is a purposely ambiguous term in this context, and could refer to region, country, state, locale, or time zone.
- **Period of time.** Start and finish date of the period. Default: forever. Period during which contributions are counted.
- **Type of contributor**, for example:
  - Repository authors
  - Issue authors
  - Code review participants
  - Mailing list authors
  - Event participants
  - IRC authors
  - Blog authors
  - By release cycle
  - Programming languages of the project
  - Role or function in project

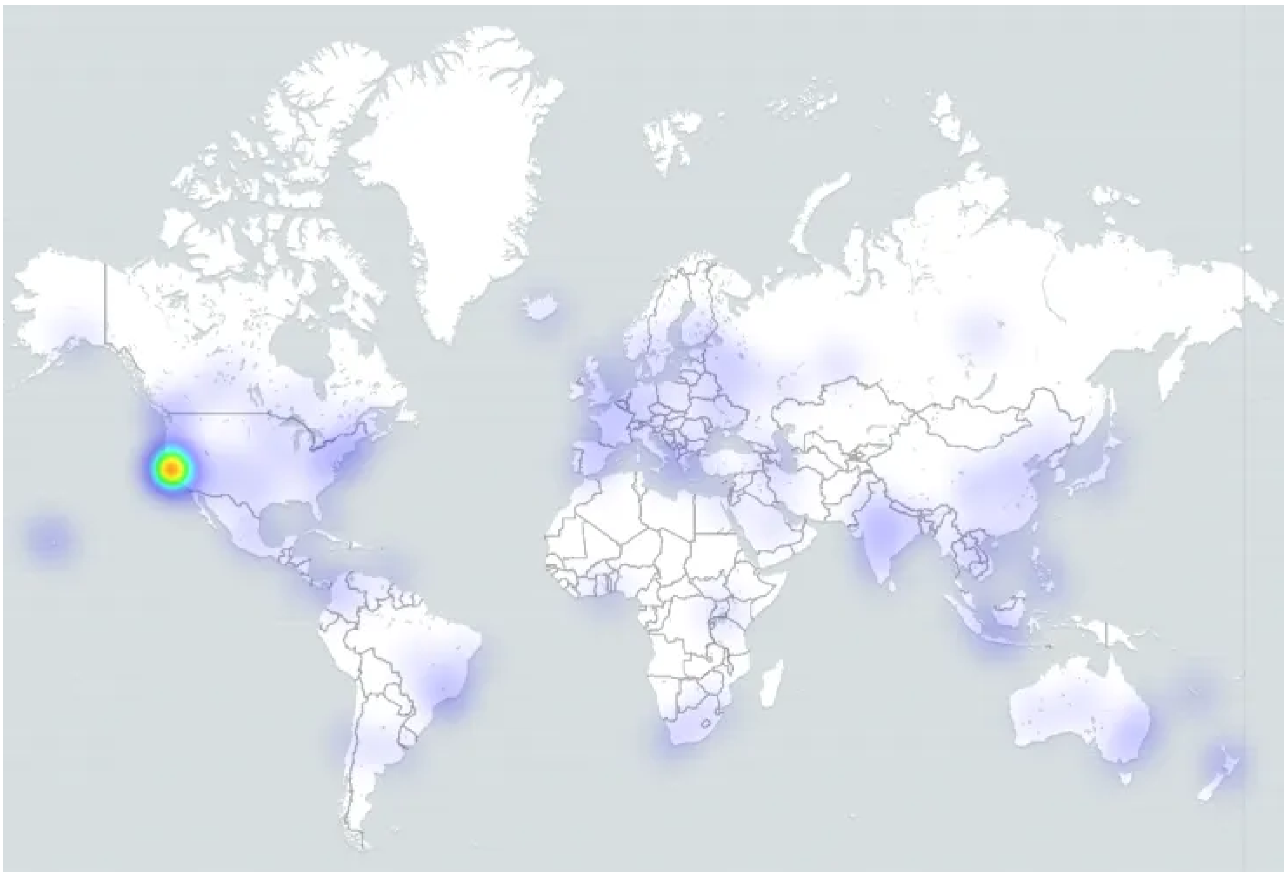
Visualizations

Dot Density Map:



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

Visual heat map:



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

Tools providing the metric

- GrimoireLab
- Augur

Data Collection Strategies

Different approaches can be used to collect information about location:

- Collect the location information from a contributor's profile in the system of engagement.
- Use IP address geolocation of the most frequent locations that contributions are made.
- Infer geographical location from the timestamp in contributions.
- Survey contributors.

The key challenge for collecting data is determining the location of the contributor. Best practice would be to leverage any profile information available from the system of engagement, and if that is not available then use IP geolocation to determine the most frequent location of contribution from that individual. Note that contributors may enter in their profile information false or nonsensical location information (e.g., “Earth” or “Internet”). Note that IP geolocation can provide large numbers of false positives due to use of VPNs or other IP masking tools.

An additional consideration would be the use of external data collection tools such as community surveys or event registration data that could cross reference systems of engagement profiles. Contributor location data could be collected inline with event attendee demographics and speaker demographics.

References

- 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.

Contributors

Question: Who are the contributors to a project?

Description

A contributor is defined as anyone who contributes to the project in any way. This metric ensures that all types of contributions are fully recognized in the project.

Objectives

Open source projects are comprised of a number of different contributors. Recognizing all contributors to a project is important in knowing who is helping with such activities as code development, event planning, and marketing efforts.

Implementation

Collect author names from collaboration tools a project uses.

Aggregators:

- Count. Total number of contributors during a given time period.

Parameters:

- Period of time. Start and finish date of the period. Default: forever. Period during which contributions are counted.

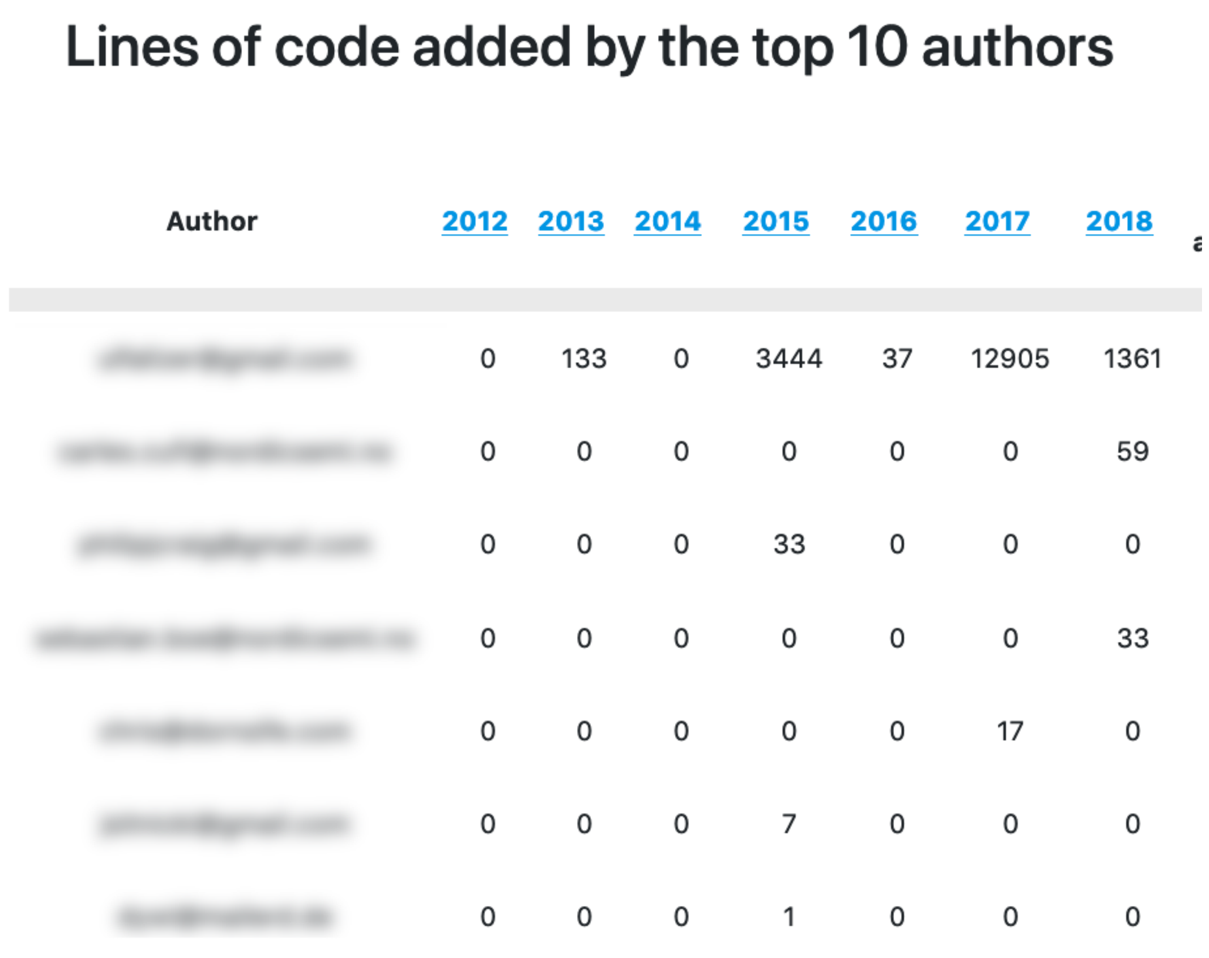
Filters

By location of engagement. For example:

- Commit authors
- Issue authors
- Review participants, e.g., in pull requests
- Mailing list authors
- Event participants
- IRC authors
- Blog authors
- By release cycle
- Timeframe of activity in the project, e.g, find new contributors
- Programming languages of the project
- Role or function in project

Visualizations

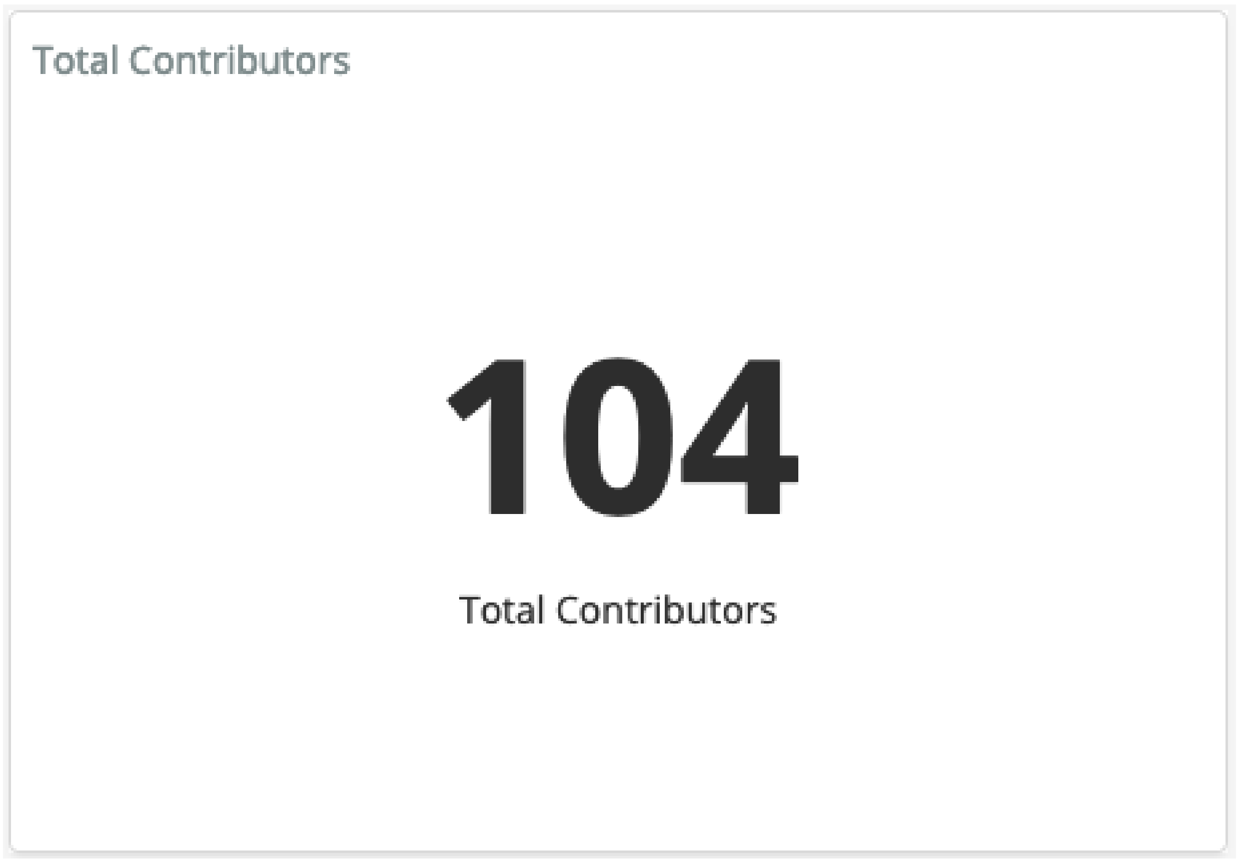
- List of contributor names (often with information about their level of engagement)



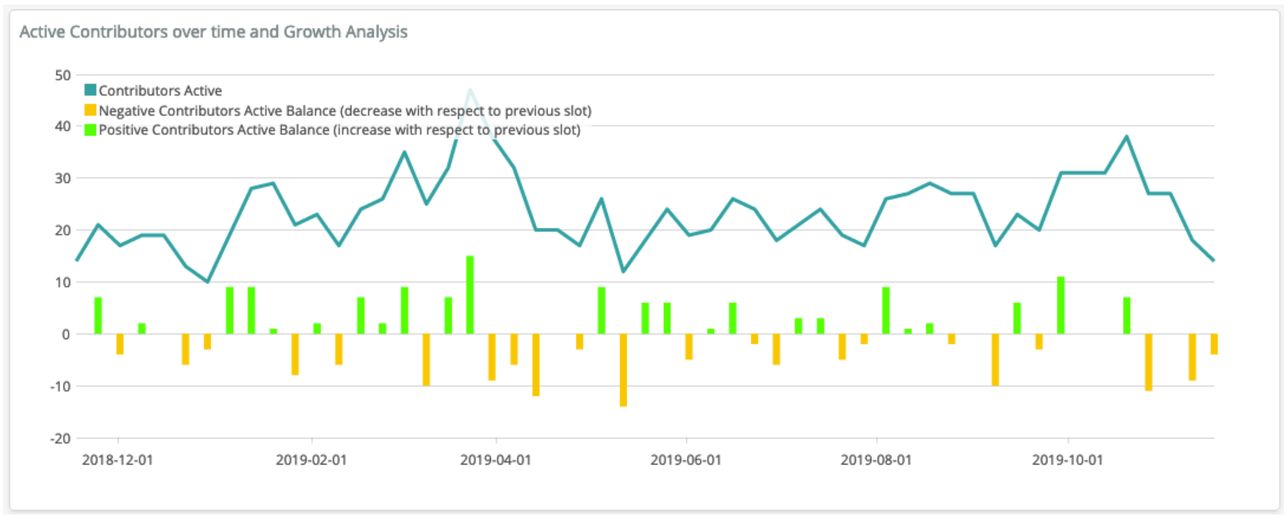
Lines of code added by the top 10 authors

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

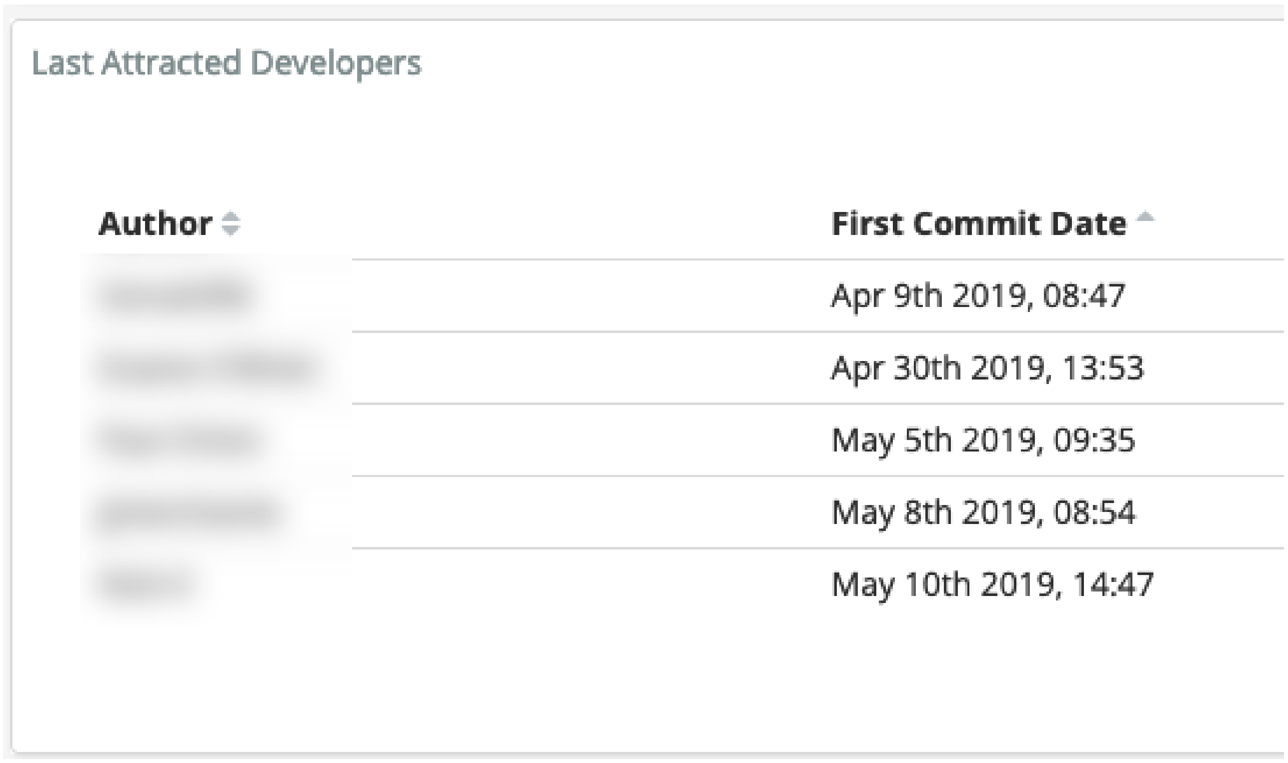
- Summary number of contributors



- Change in the number of active contributors over time



- New contributors (sort list of contributors by date of first contribution)


Last Attracted Developers

Author	First Commit Date
[redacted]	Apr 9th 2019, 08:47
[redacted]	Apr 30th 2019, 13:53
[redacted]	May 5th 2019, 09:35
[redacted]	May 8th 2019, 08:54
[redacted]	May 10th 2019, 14:47

Tools Providing the Metric

- GrimoireLab
- Augur

Data Collection Strategies

As indicated above, some contributor information is available via software such as GrimoireLab and Augur. However, some contributor insights are less easily obtained via trace data. In these cases, surveys with community members or event registrations can provide the desired information. Sample questions include:

- Interview question: Which contributors do not typically appear in lists of contributors?
- Interview question: Which contributors are often overlooked as important contributors because their contributions are more “behind the scenes”?
- Interview question: What other community members do you regularly work with?

Additionally, surveys with community members can provide insight to learn more about contributions to the project. Sample questions include:

- Likert scale [1-x] item: I am contributing to the project
- Matrix survey item: How often do you engage in the following activities in the project?
  - Column headings: Never, Rarely(less than once a month), Sometimes (more than once a month), Often(once a week or more)
  - Rows include: a) Contributing/reviewing code, b) Creating or maintaining documentation, c) Translating documentation, d) Participating in decision making about the project’s development, e) Serving as a community organizer, f) Mentoring other contributors, g) Attending events in person, h) Participating through school or university computing programs, i) Participating through a program like Outreachy, Google Summer of Code, etc., j) Helping with the ASF operations (e.g., board meetings or fundraising)

References

Organizational Diversity

Question: What is the organizational diversity of contributions?

Description

Organizational diversity expresses how many different organizations are involved in a project and how involved different organizations are compared to one another.

Objectives

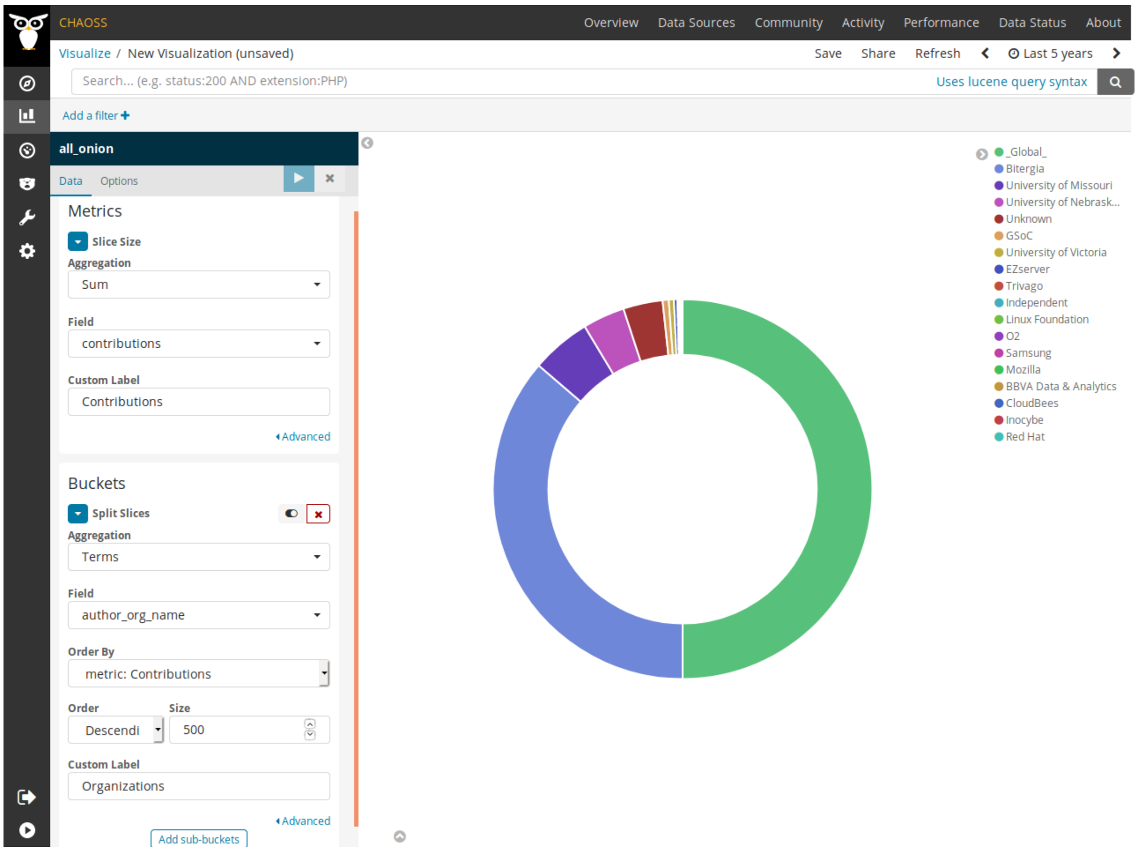
- Get a list of organizations contributing to a project.
- See the percentage of contributions from each organization within a defined period of time.
- See the change of composition of organizations within a defined period of time.
- Get a list of people that are associated with each organization.

Implementation

- Collect data from data sources where contributions occur.
- Identify contributor affiliations to get a good estimate of which organizations they belong to.
- Correlate information about contributions, assigning each to appropriate organization.
- Depending on the needs of the project, you may want to consider such issues as how to handle multiple email addresses, affiliation changes over time, or contractor vs. employee.

Tools Providing the Metric

- GrimoireLab supports organizational diversity metrics out of the box. The GrimoireLab SortingHat manages identities. The GrimoireLab Hatstall user interface allows correcting organizational affiliation of people and even recording affiliation changes.
  - View an example visualization on the CHAOSS instance of Bitergia Analytics.
  - Download and import a ready-to-go dashboard containing examples for this metric visualization from the GrimoireLab Sigils panel collection.
  - Add a sample visualization to any GrimoreLab Kibiter dashboard following these instructions:
    - Create a new Pie chart
      - Select the `all_onion` index
      - Metrics Slice Size: `Sum` Aggregation, `contributions` Field, `Contributions` Custom Label
      - Buckets Split Slices: `Terms` Aggregation, `author_or_name` Field, `metric: Contributions` Order By, `Descending` Order, `500` Size, `Organization` Custom Label
    - Example Screenshot



Visualize / New Visualization (unsaved)

Search... (e.g. status:200 AND extension:PHP)

all_onion

Metrics

Slice Size

Aggregation: Sum

Field: contributions

Custom Label: Contributions

Advanced

Buckets

Split Slices

Aggregation: Terms

Field: author_org_name

Order By: metric: Contributions

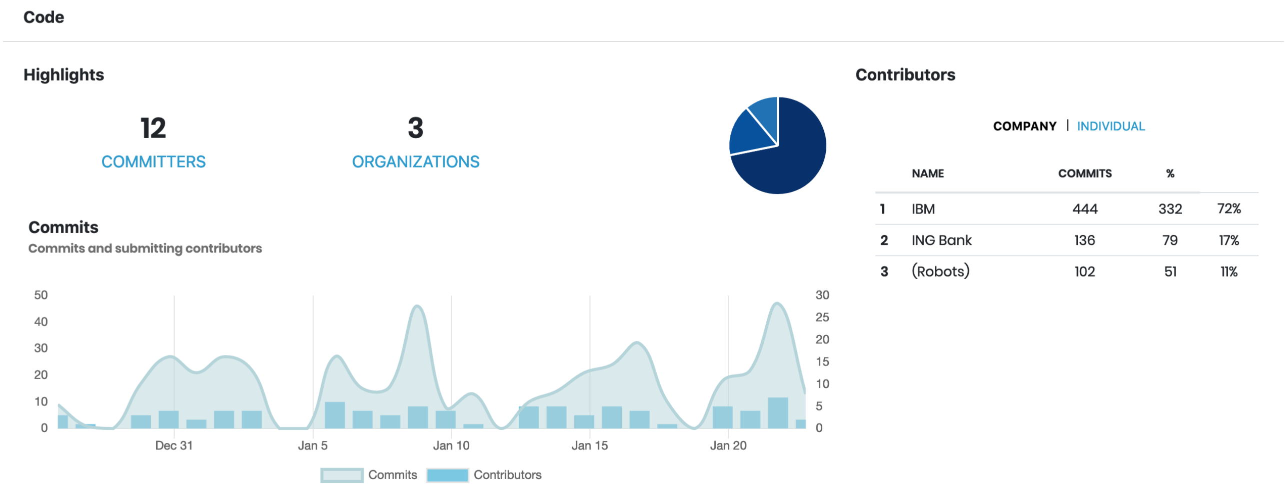
Order: Descendi

Size: 500

Custom Label: Organizations

Advanced

- LF Analytics provides organization diversity metrics in the primary view for commits, issues filed, and communication channels (current support for Slack and groups.io)



Code

Highlights

12 COMMITTERS

3 ORGANIZATIONS

Commits

Commits and submitting contributors

Contributors

COMPANY | INDIVIDUAL

	NAME	COMMITS		%
1	IBM	444	332	72%
2	ING Bank	136	79	17%
3	(Robots)	102	51	11%

Data Collection Strategies

Qualitative

- Footprint of an organization in a project or ecosystem
- Influence of an organization in a project or ecosystem
- Affiliation diversity in governance structures.

Quantitative

- % of commits by each organization
- % of merges/reviews from each organization

- % of any kind of contributors from each organization
- % of lines of code contributed by each organization
- % issues filed by each organization
- Contributing Organizations - What is the number of contributing organizations?
- New Contributing Organizations - What is the number of new contributing organizations?
- New Contributor Organizations - New organizations contributing to the project over time.
- Number of Contributing Organizations - Number of organizations participating in the project over time.
- Elephant Factor - If 50% of community members are employed by the same company, it is the elephant in the room. Formally: The minimum number of companies whose employees perform 50% of the commits
- Affiliation Diversity - Ratio of contributors from a single company over all contributors. Also described as: Maintainers from different companies. Diversity of contributor affiliation.
- In projects with the concept of code ownership, % of code owners affiliated with each organization weighed by the importance/size/LoC of the code they own and the number of co-owners.

References

- Potential implementations and references:
  - https://bitergia.gitlab.io/panel-collections/open_source_program_office/organizational-diversity.html
  - Kata Containers dashboard entry page (bottom of this)
  - Augur