GitClear

A visual guide to

Diff Delta

And how fusing commits together reduces tech debt & code review time

GitClear

GitClear provides an engineering insights tool built **to create incentives to reduce tech debt and ship faster**. We do it by identifying the rate at which code evolves: per-developer, per-directory, and per-team.

We want developers to enjoy reviewing code -- like it's a good book. By making code easier to read, we help them reduce (or even prevent) the tech debt that saps developer enthusiasm on so many large projects.

GitClear went live to customers in 2019 and has been used by hundreds of teams to improve their code quality & velocity.

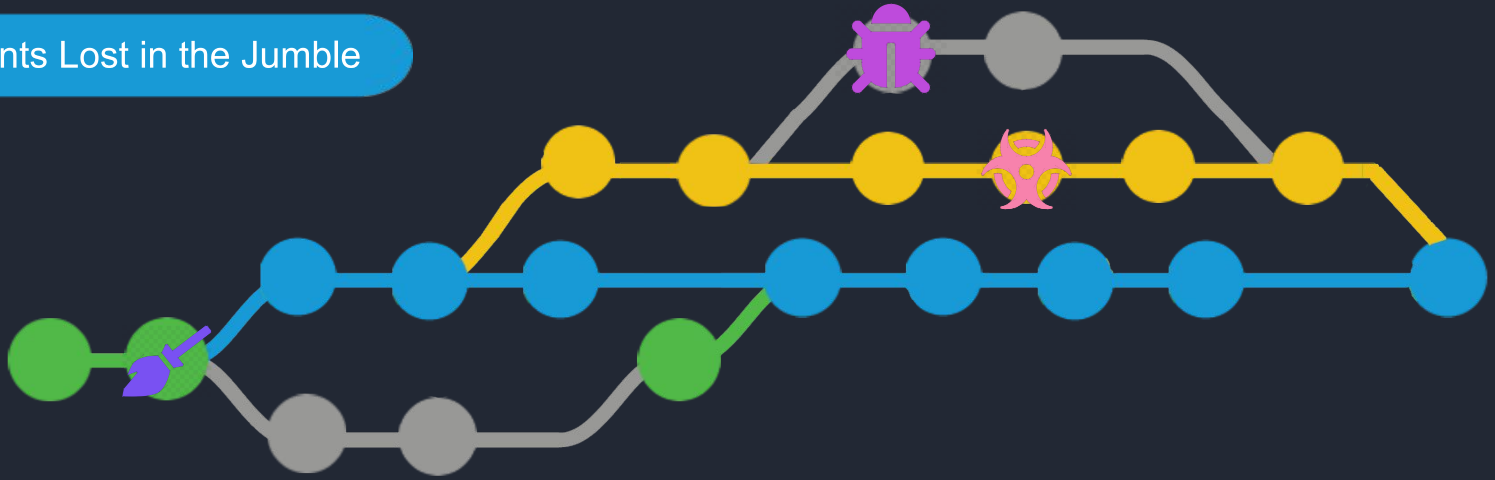
Watch a 3 minute GitClear explainer video.

The Average Git Repo in 2021

A development team might have 100+ commits made across their git repos every day. Nobody can keep track of everything. The average repo in 2021 is a black box that periodically kicks out PRs when it’s working right.

At best, a couple developers know how the big picture connects.

Critical Moments Lost in the Jumble



Somewhere in this commit jumble...

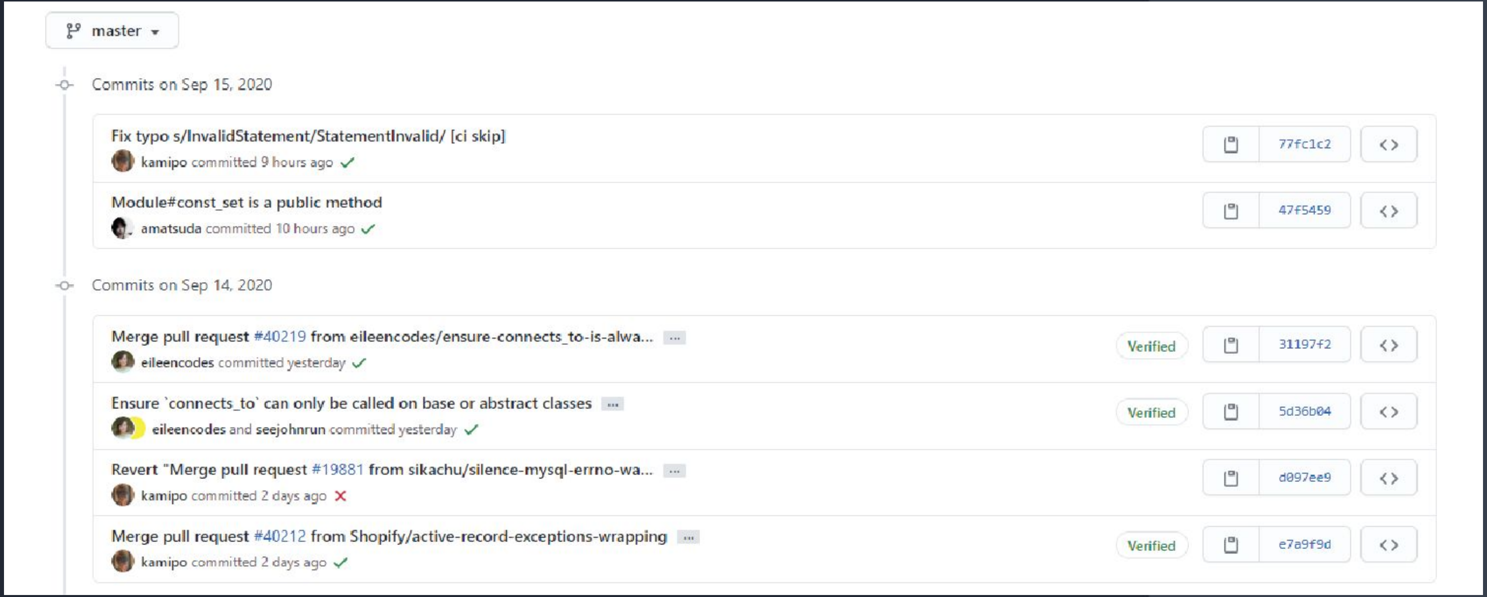
- A new bug is being introduced
- A developer is refactoring legacy code that will prevent tremendous headaches down the road
- A junior developer is creating tech debt by duplicating an existing utility function

When you’re lucky, you find out about these issues before your customers do. But just as often, critical moments happen and nobody knows about it until weeks or months later, if at all.

Where Do We Look for Critical Commits?

Certainly not in the commit list

You're not going to catch critical moments in a flat list of hundreds of commits, separated by repo. Nobody in their right mind will use this to browse commit activity.



master

Commits on Sep 15, 2020

Fix typo s/InvalidStatement/StatementInvalid/ [ci skip]
kamipo committed 9 hours ago ✓ — 77fc1c2

Module#const_set is a public method
amatsuda committed 10 hours ago ✓ — 47f5459

Commits on Sep 14, 2020

Merge pull request #40219 from eileencodes/ensure-connects_to-is-alwa... …
eileencodes committed yesterday ✓ — Verified — 31197f2

Ensure `connects_to` can only be called on base or abstract classes …
eileencodes and seejohnrun committed yesterday ✓ — Verified — 5d36b04

Revert "Merge pull request #19881 from sikachu/silence-mysql-errno-wa... …
kamipo committed 2 days ago ✗ — d097ee9

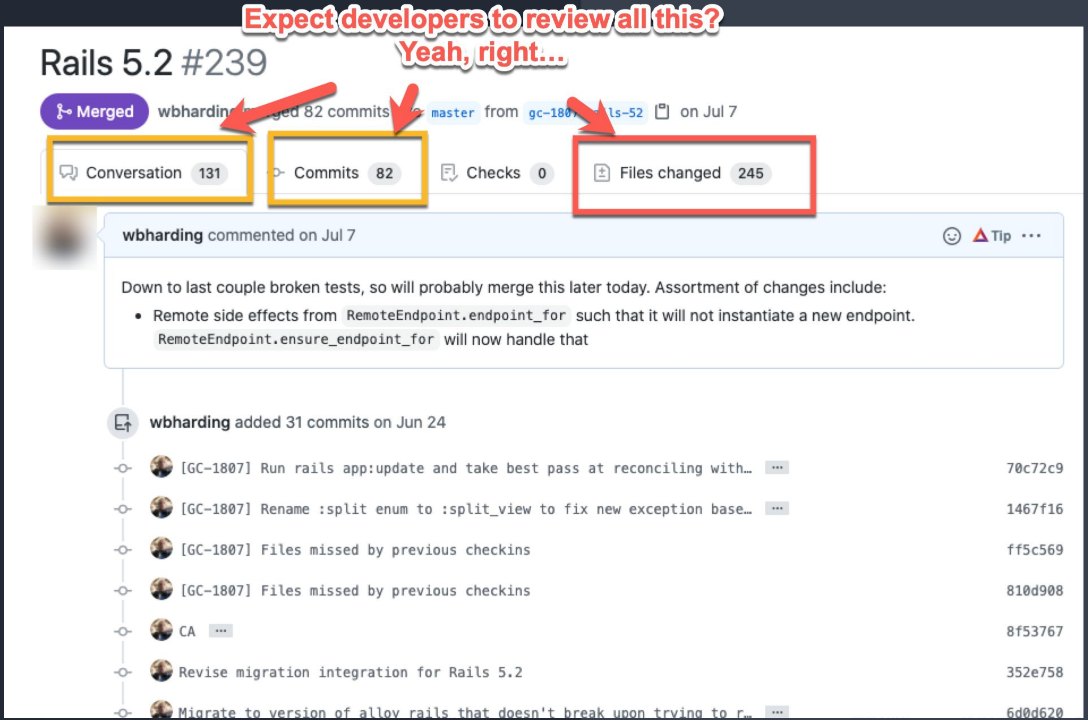
Merge pull request #40212 from Shopify/active-record-exceptions-wrapping …
kamipo committed 2 days ago ✓ — Verified — e7a9f9d

Where Do We Look for Critical Commits?

Not in PRs

PRs are *supposed* to clean up the mess that happens in development, but you've seen how that goes...

- **The change is too overwhelming**. Unless your team always keeps its PRs below 20 commits, they become overwhelming and are glossed over
- **Bugs and tech debt abound by the time PR stage reached**. Especially for new team members or Junior Developers, when weeks may pass between PRs.
- **The top people don't have time to read them.** Senior Architects are too busy to read PRs. If it's small, expect a quick glance. If it's more than 20 commits, you don't want to know...
- **On small teams, much work happens outside of the PR process**. Such work usually gets reviewed by no one.



Rails 5.2 #239

Merged wbhardin ged 82 commits master from gc-180 ls-52 on Jul 7

Conversation 131 | Commits 82 | Checks 0 | Files changed 245

wbharding commented on Jul 7

Down to last couple broken tests, so will probably merge this later today. Assortment of changes include:

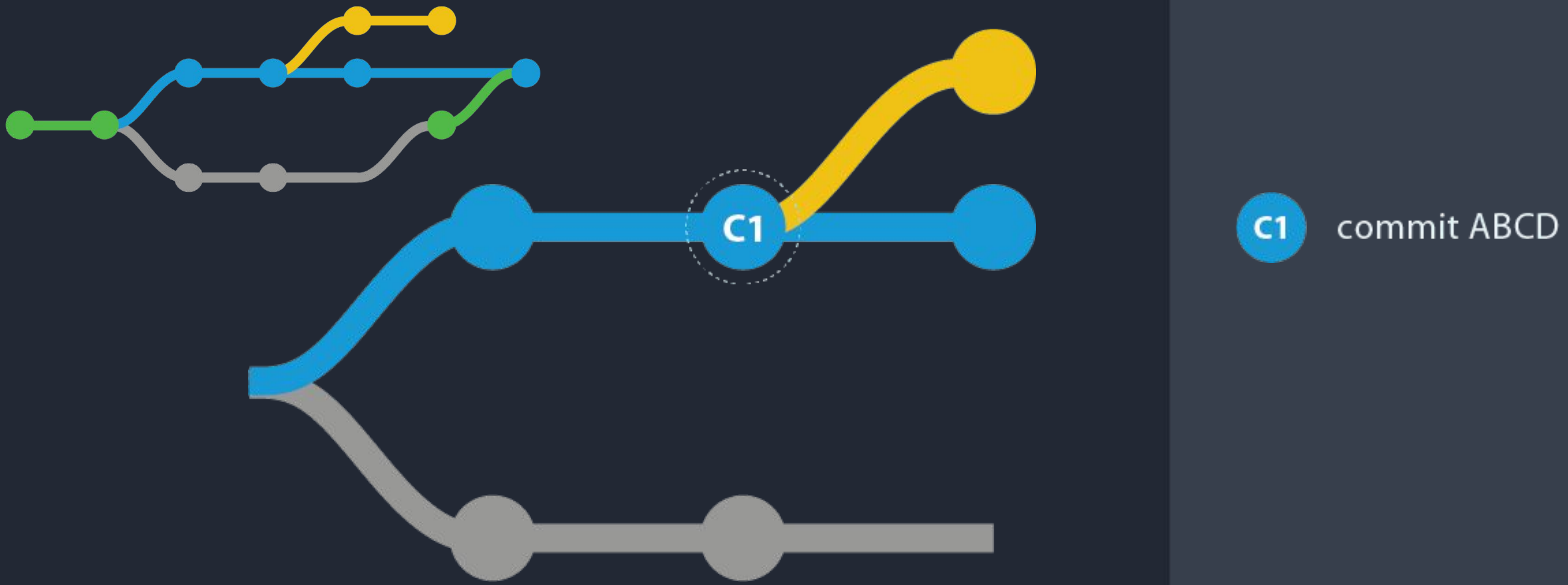
- Remote side effects from `RemoteEndpoint.endpoint_for` such that it will not instantiate a new endpoint. `RemoteEndpoint.ensure_endpoint_for` will now handle that

wbharding added 31 commits on Jun 24

```
[GC-1807] Run rails app:update and take best pass at reconciling with… …        70c72c9
[GC-1807] Rename :split enum to :split_view to fix new exception base… …        1467f16
[GC-1807] Files missed by previous checkins                                     ff5c569
[GC-1807] Files missed by previous checkins                                     810d908
CA …                                                                            8f53767
Revise migration integration for Rails 5.2                                      352e758
Migrate to version of alloy rails that doesn't break upon trying to r… …        6d0d620
```

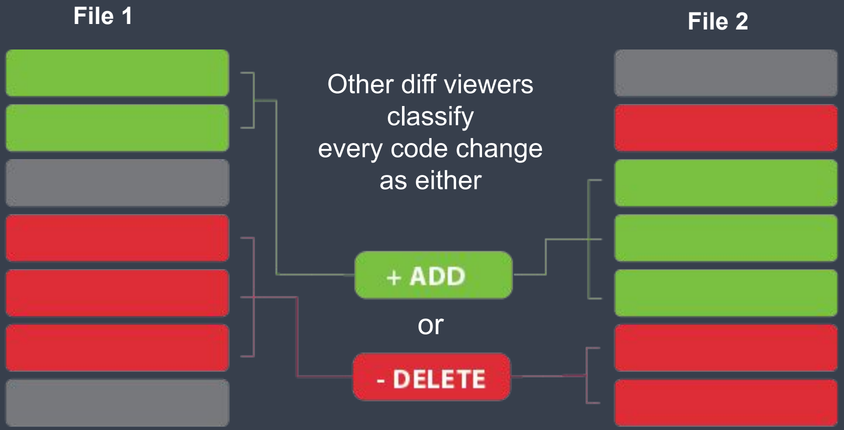
GitClear Opening the “Repo Black Box”

Let’s zoom into a single commit to see how GitClear’s rich commit parsing engine uses Diff Delta to simplify even the most intricate commit activity



The Old Way: Binary Diff

- When git tools like Github interpret a commit, they relate a list of green and red changes. **Everything is binary.**

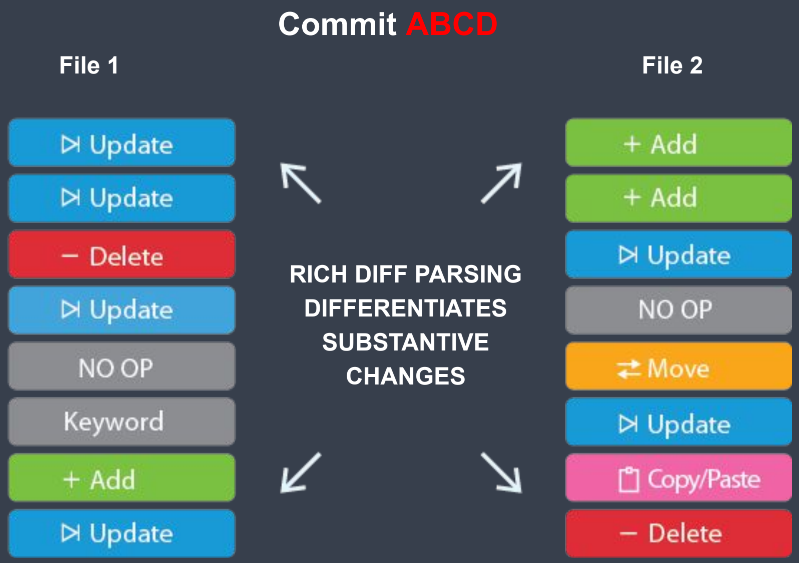


By treating every change as either an “add” or “delete,” other git tools put the onus on code reviewers to piece together the real story. Is an **apparent block of "added code" is truly new, or just some legacy code moved in from another file?**

Since 30% of all line changes are moved code, developers could be 30% faster understanding code by using a more refined diff tool.

The GitClear Way: Diff Like a Developer

Meanwhile, GitClear recognizes...



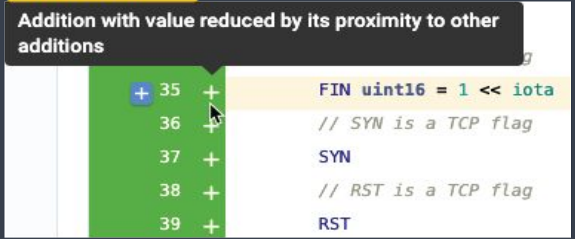
By classifying code using the full set of code operations that developers recognize, we can extract semantic meaning to differentiate between big vs trivial changes

Technical Details Optional In-Depth Reading

Operations Recognized by Diff Delta

Here are seven types of operations we recognize in commits. Each operation is accompanied by a screenshot of how the operation looks in the GitClear diff viewer.

1. Addition

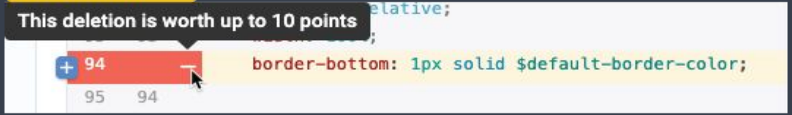


Addition with value reduced by its proximity to other additions

```
35 +    FIN uint16 = 1 << iota
36 +    // SYN is a TCP flag
37 +    SYN
38 +    // RST is a TCP flag
39 +    RST
```

Each added line of code counts for up to 10 points.

2. Deletion

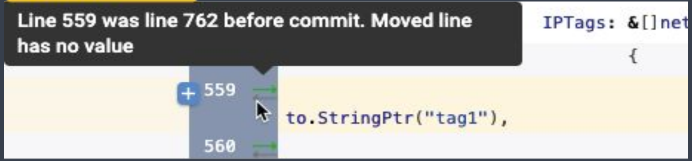


This deletion is worth up to 10 points

```
94     -    border-bottom: 1px solid $default-border-color;
95  94
```

Each deleted line of code can count for up to 25 points.
By default, Diff Delta prizes code deletion most of all, for its role in reducing long-term tech debt.

3. Move

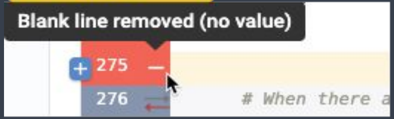


Line 559 was line 762 before commit. Moved line has no value

```
IPTags: &[]net
    {
559     to.StringPtr("tag1"),
560
```

Moved code (about 30% of all changed lines) is assigned no Diff Delta

4. No-ops



Blank line removed (no value)

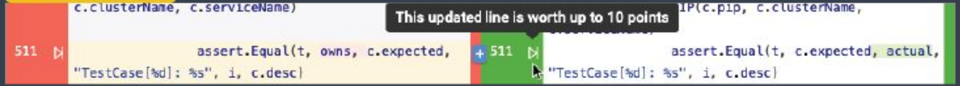
```
275 -
276       # When there a
```

One of the most common types of code change is the "no-op." This encompasses all changes to white space, blank lines added, and lines whose only change was their line number.

Technical Details Optional In-Depth Reading

Operations Recognized by Diff Delta

5. Update

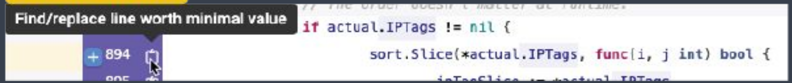


```
c.clusterName, c.serviceName)
            assert.Equal(t, owns, c.expected,
"TestCase[%d]: %s", i, c.desc)
```

```
IP(c.pip, c.clusterName,
            assert.Equal(t, c.expected, actual,
"TestCase[%d]: %s", i, c.desc)
```

When a line changes in part, we consider this an "update." Updates can count for up to 10 points.

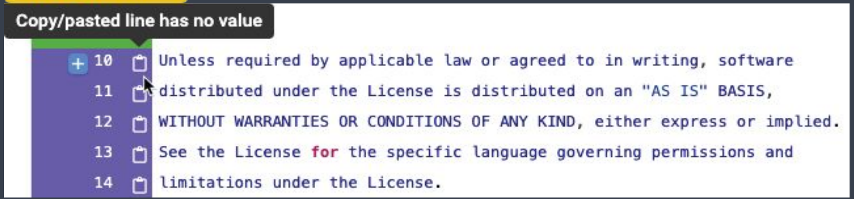
6. Find & replace



```
if actual.IPTags != nil {
        sort.Slice(*actual.IPTags, func(i, j int) bool {
```

When a developer applies the same change to several lines en masse, this is detected as "Find & replace." Such lines are worth up to 3 points.

7. Copy/Paste



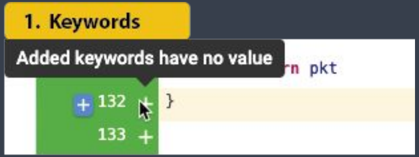
```
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

When a developer repeatedly adds ("pastes") the same line or block in multiple locations, across one or more commits. Copy/pasted code is assigned no Diff Delta for its role in creating tech debt.

Technical Details Optional In-Depth Reading

Language Idioms Recognized by Diff Delta

Assuming you're using one of the 40 programming languages (including every modern language), Diff Delta will know a few more tricks out of the box. Below, a subset of language-specific idioms we recognize.

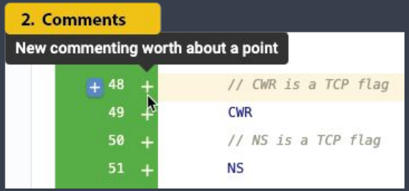


1. Keywords

Added keywords have no value

```
rn pkt
}
```

About 10% of lines changes are language keywords: these are transparent to Diff Delta.

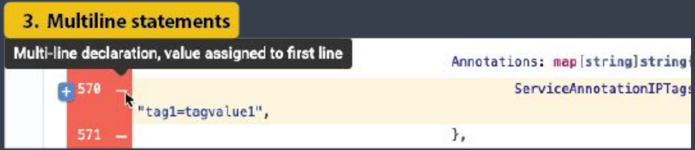


2. Comments

New commenting worth about a point

```
// CWR is a TCP flag
CWR
// NS is a TCP flag
NS
```

Comments are assigned negligible Diff Delta

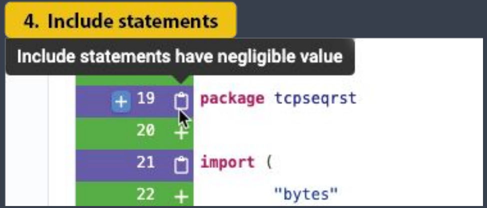


3. Multiline statements

Multi-line declaration, value assigned to first line

```
                Annotations: map[string]string
                    ServiceAnnotationIPTags
"tag1=tagvalue1",
                },
```

Multi-line statements have their value assigned to the first line



4. Include statements

Include statements have negligible value

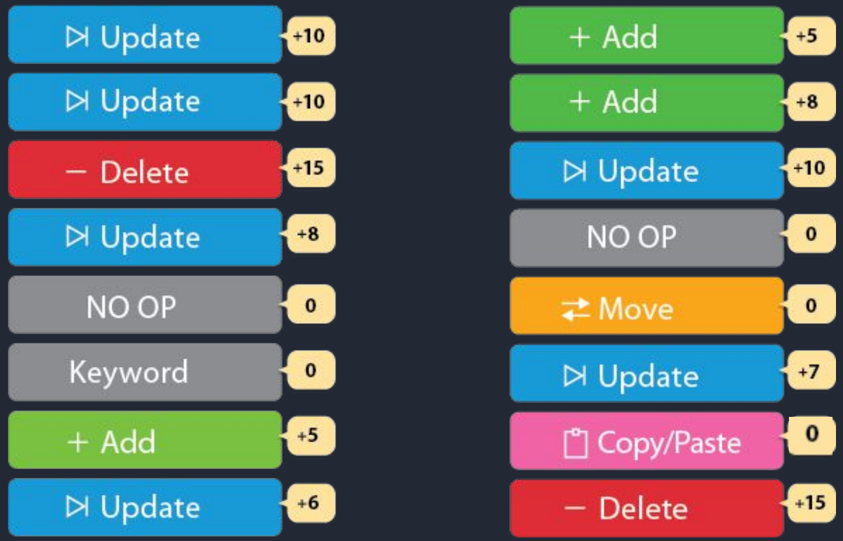
```
package tcpseqrst

import (
        "bytes"
```

“Include statements” can comprise up to 5% of changed lines in React repos. Diff Delta treats them as trivial changes.

Diff Delta ≈ Cognitive energy to make a change

Initial estimation phase

Commit ABCD



File 1		File 2	
⋈ Update	+10	+ Add	+5
⋈ Update	+10	+ Add	+8
− Delete	+15	⋈ Update	+10
⋈ Update	+8	NO OP	0
NO OP	0	⇄ Move	0
Keyword	0	⋈ Update	+7
+ Add	+5	Copy/Paste	0
⋈ Update	+6	− Delete	+15

+100 Diff Delta

Initial Estimate

Once our commit parse is complete, we assign a provisional score based on the estimated cognitive energy needed to produce the changes in the commit [1].

Scores range from 0-30 per line, with the highest scores reserved for updates and deletions to legacy code, where tech debt tends to accumulate and only experts dare tread.

Changes like no-ops, moved code, and copy/paste get no Diff Delta. Most non-trivial changes get 5-10 per changed line. New code earns little value since it is prone to churn and begets maintainability costs.

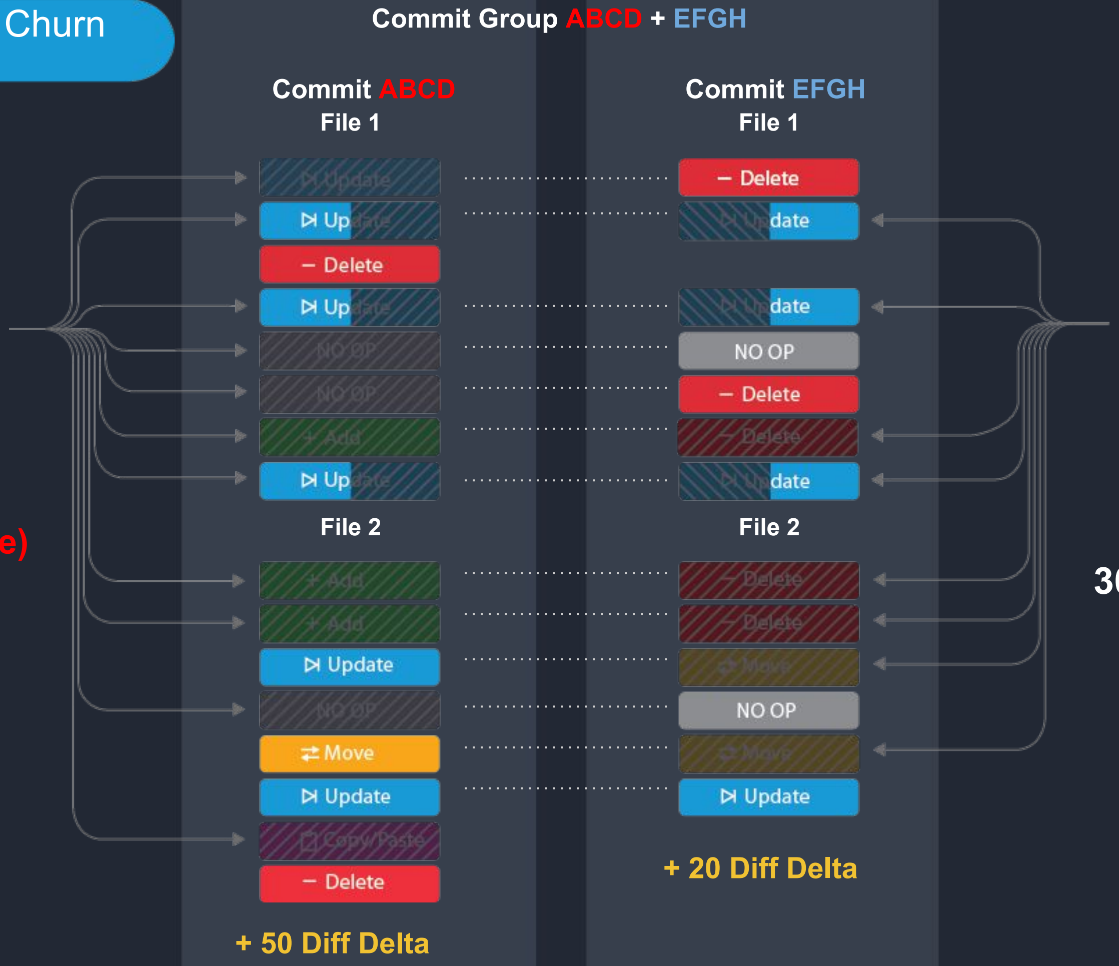
[1] https://www.gitclear.com/diff_delta_factors

Diff Delta: Incorporating Churn

Secondary refinement phase

WORK RENDERED OBSOLETE by Commit EFGH

100 - 50 (obsolete) Diff Delta = 50 New Delta



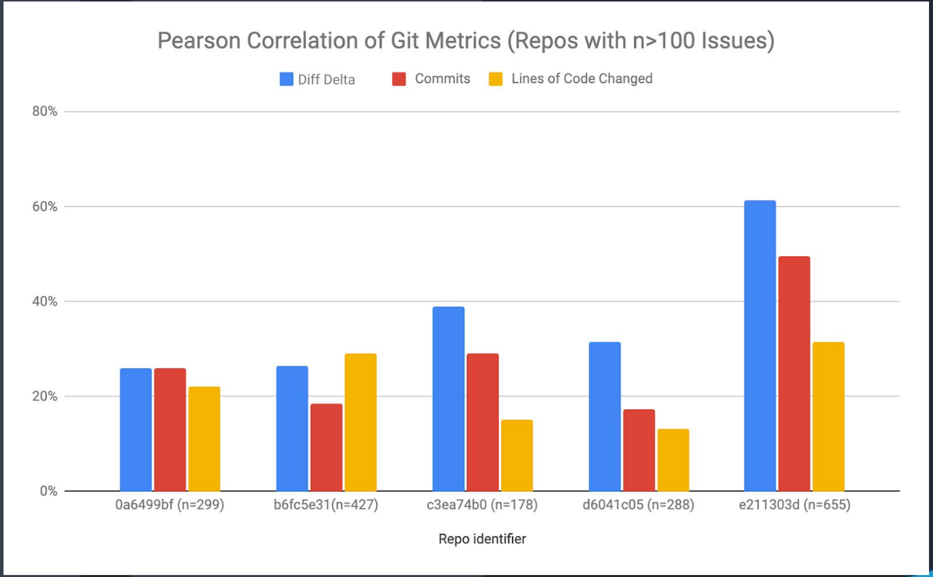
WORK RENDERED OBSOLETE by Commit ABCD

30 - 10 (obsolete) Diff Delta = 20 New Delta

Can Diff Delta be Trusted?

To evaluate whether Diff Delta could empirically outperform other common git metrics, GitClear collected 2,800 data points to compare how well Diff Delta correlated to “developer effort.” We found “Diff Delta” correlated up to 61% in large repos with effort. Read what academics had to say about our research.

At right, the blue bar indicates the degree of correlation between Diff Delta and effort spend by the development team. Diff Delta matches effort better than conventional metrics “commits” or “lines of code.”

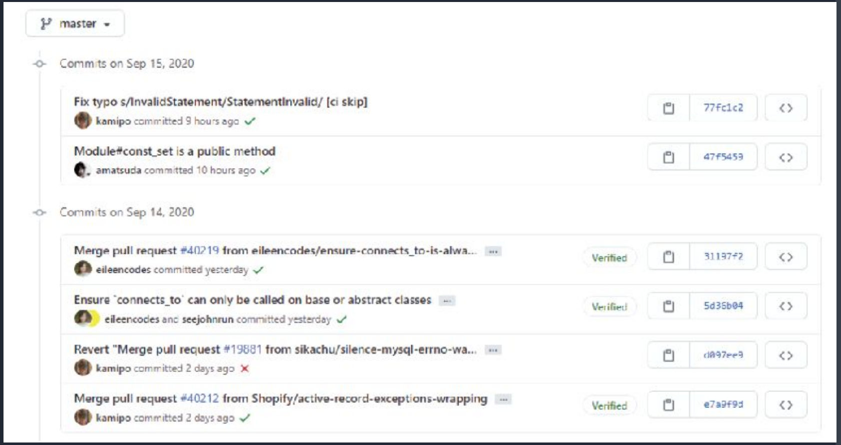


Commit Activity Browser: 2d Visual Map

GitClear rearranges the flat list of commits you're used to into a colorful, 2d browsable map of commits

Commits are grouped by issue and condensed to hide irrelevant changes

Learn more about Commit Activity Browser


master

Commits on Sep 15, 2020

Fix typo s/InvalidStatement/StatementInvalid/ [ci skip]
kamipo committed 9 hours ago ✓ — 77fc1c2

Module#const_set is a public method
amatsuda committed 10 hours ago ✓ — 47f5453

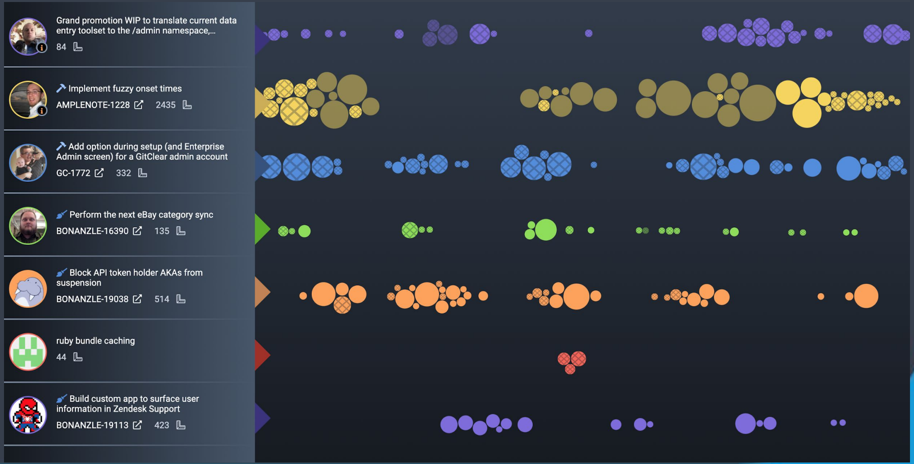
Commits on Sep 14, 2020

Merge pull request #40219 from eileencodes/ensure-connects_to-is-alwa... — Verified — 31197f2
eileencodes committed yesterday ✓

Ensure `connects_to` can only be called on base or abstract classes — Verified — 5d35b04
eileencodes and seejohnrun committed yesterday ✓

Revert "Merge pull request #19881 from sikachu/silence-mysql-errno-wa... — d097ee9
kamipo committed 2 days ago ✗

Merge pull request #40212 from Shopify/active-record-exceptions-wrapping — Verified — e7a9f9d
kamipo committed 2 days ago ✓


Grand promotion WIP to translate current data entry toolset to the /admin namespace,...
84

Implement fuzzy onset times
AMPLENOTE-1228 2435

Add option during setup (and Enterprise Admin screen) for a GitClear admin account
GC-1772 332

Perform the next eBay category sync
BONANZLE-16390 135

Block API token holder AKAs from suspension
BONANZLE-19038 514

ruby bundle caching
44

Build custom app to surface user information in Zendesk Support
BONANZLE-19113 423

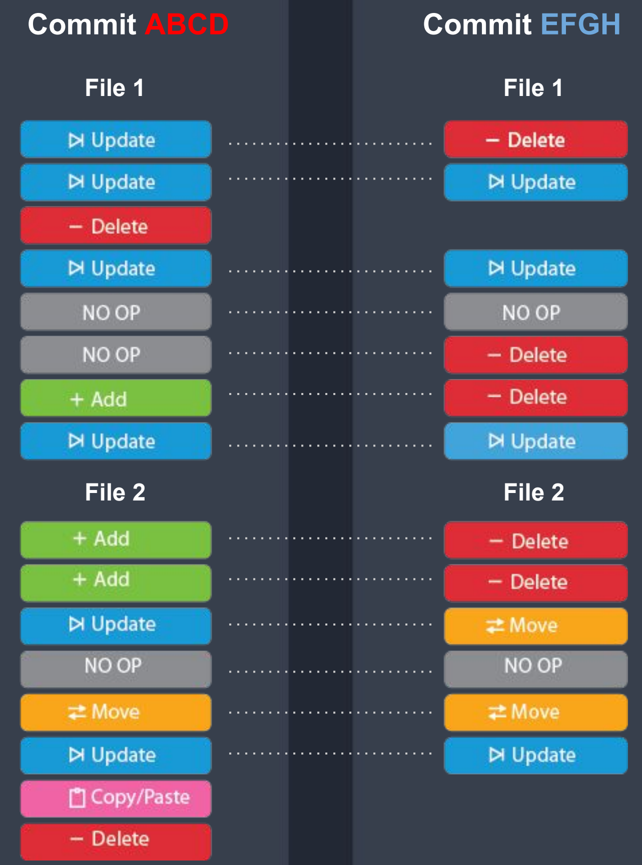
Commit Groups Expose Critical Moments

What Happens by Fusing Commits ABCD + EFGH?

In the real world, commits don't happen in isolation. Diff Delta is built to get more accurate as more commits are added.

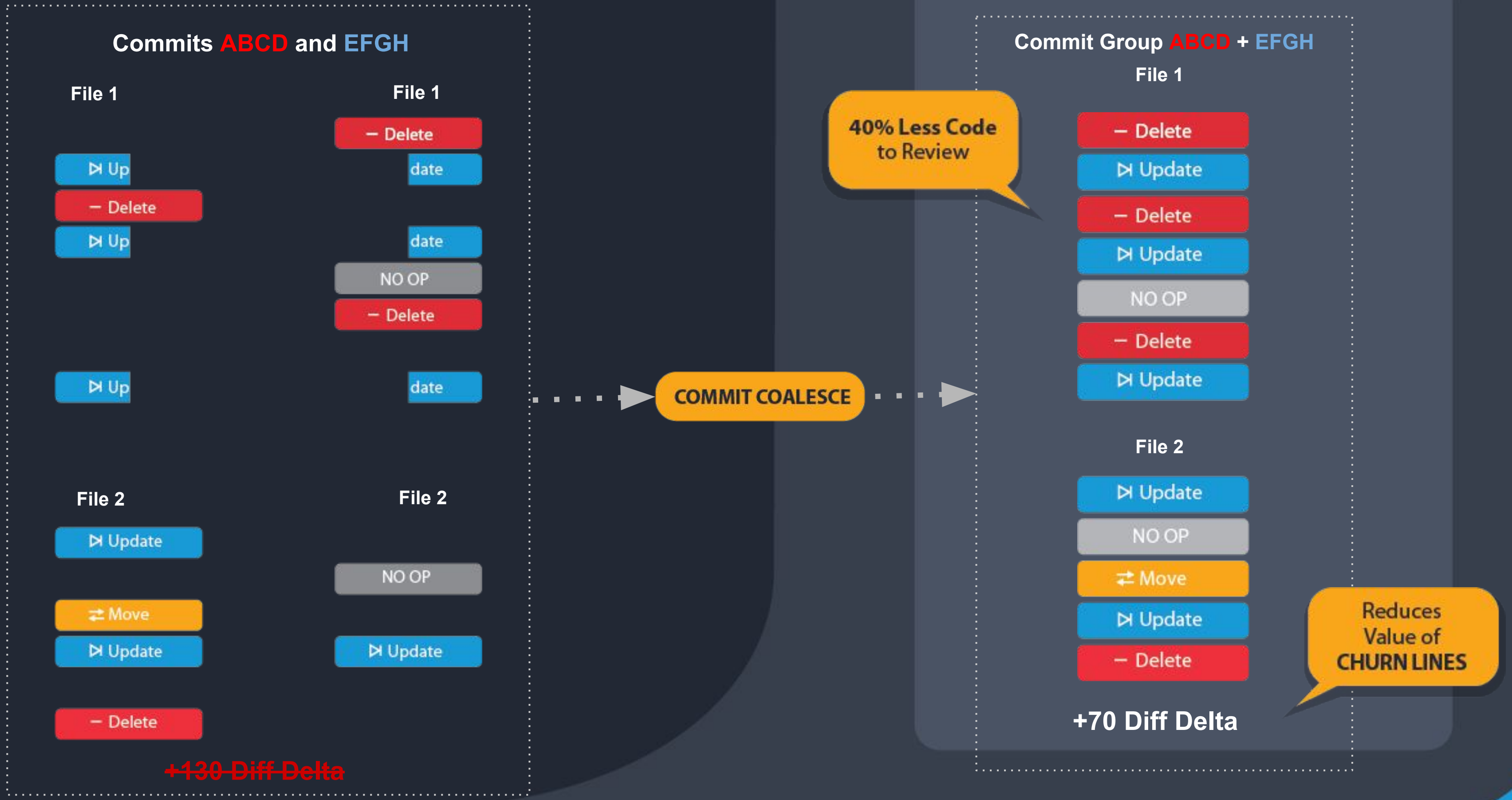
By fusing together related commits into Commit Groups, we cancel out noise from churn and get the ground truth about what precisely changed over the course of an issue's implementation.

Fusing commits into commit groups is key to spotting bugs before they reach production.

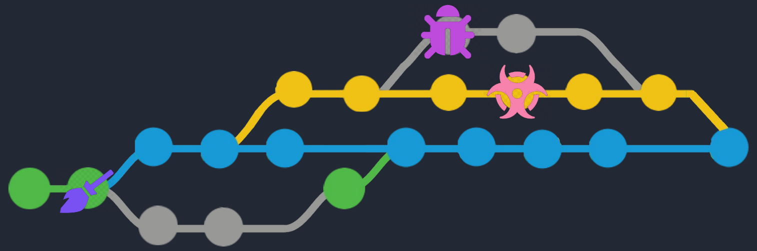


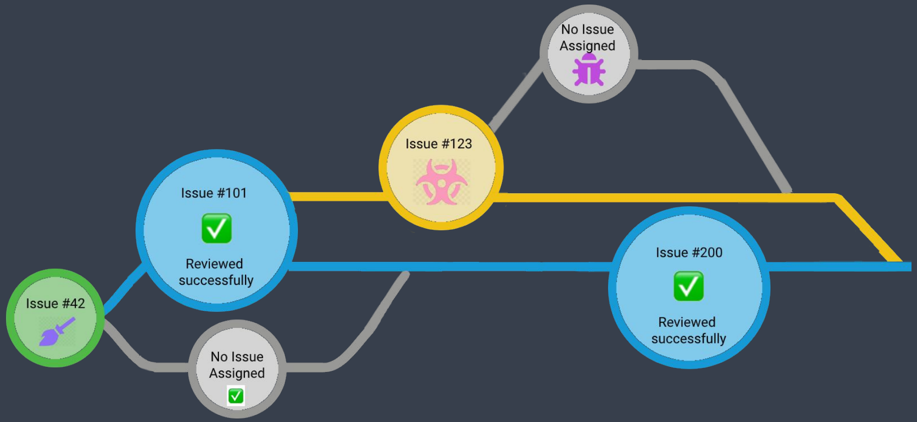
Technical Details

Making a Commit Group



The Difference is (Git)Clear





Without GitClear:

- *Bugs are created and nobody knows about it*
- *Developers interrupted to get updates*
- *Junior Developers repeat same mistakes*

With GitClear:

★ *Code and Jira together from first commit*
★ *Real-time developer status, no interruption*
★ *Team crafts more durable code together*

Don’t Take Our Word for It...

Thousands of startup and enterprise teams trust GitClear to make sense of their development activity

fff

Trusted By

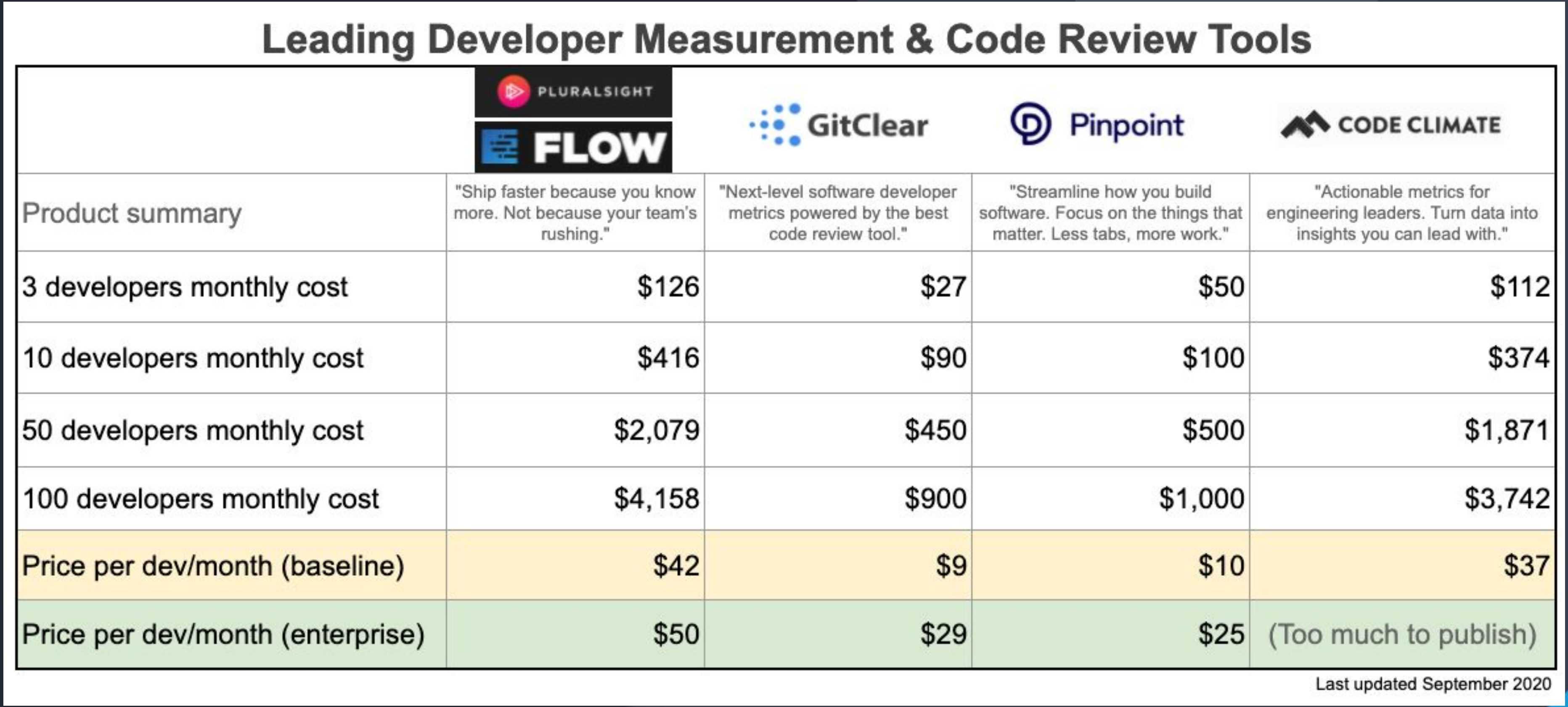
nextgen® healthcare

one One Network Enterprises™

HCSC Health Care Service Corporation

Competitive Pricing for Every Size Team

Try GitClear free



Leading Developer Measurement & Code Review Tools

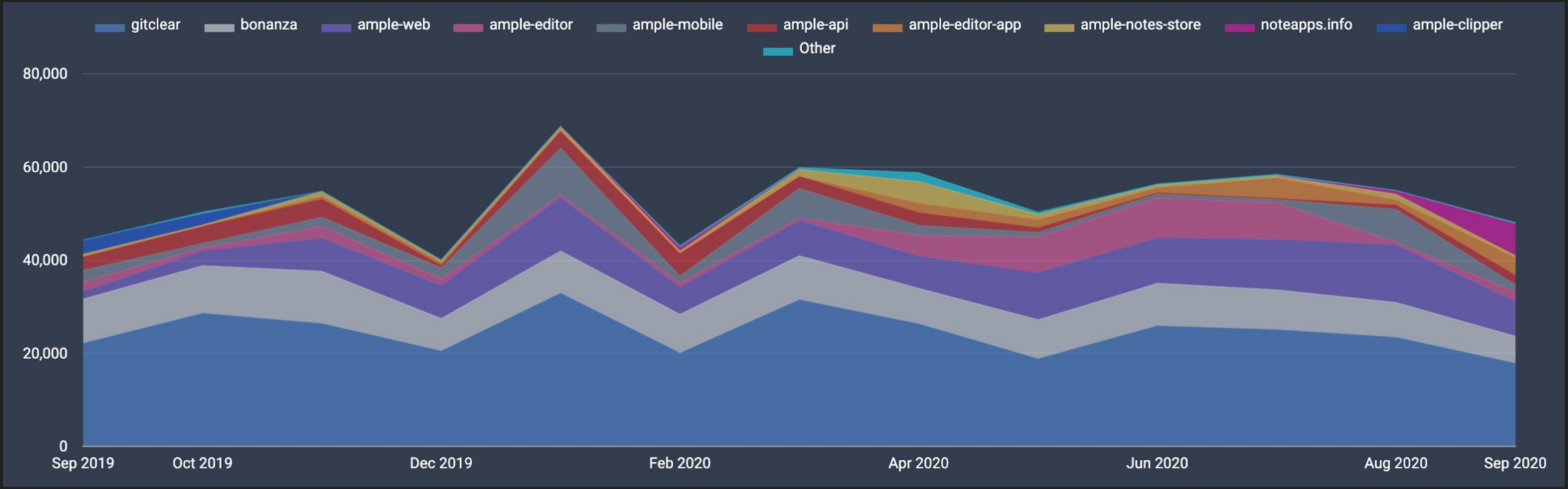
	PLURALSIGHT FLOW	GitClear	Pinpoint	CODE CLIMATE
Product summary	"Ship faster because you know more. Not because your team's rushing."	"Next-level software developer metrics powered by the best code review tool."	"Streamline how you build software. Focus on the things that matter. Less tabs, more work."	"Actionable metrics for engineering leaders. Turn data into insights you can lead with."
3 developers monthly cost	$126	$27	$50	$112
10 developers monthly cost	$416	$90	$100	$374
50 developers monthly cost	$2,079	$450	$500	$1,871
100 developers monthly cost	$4,158	$900	$1,000	$3,742
Price per dev/month (baseline)	$42	$9	$10	$37
Price per dev/month (enterprise)	$50	$29	$25	(Too much to publish)

Last updated September 2020

Bonus Content

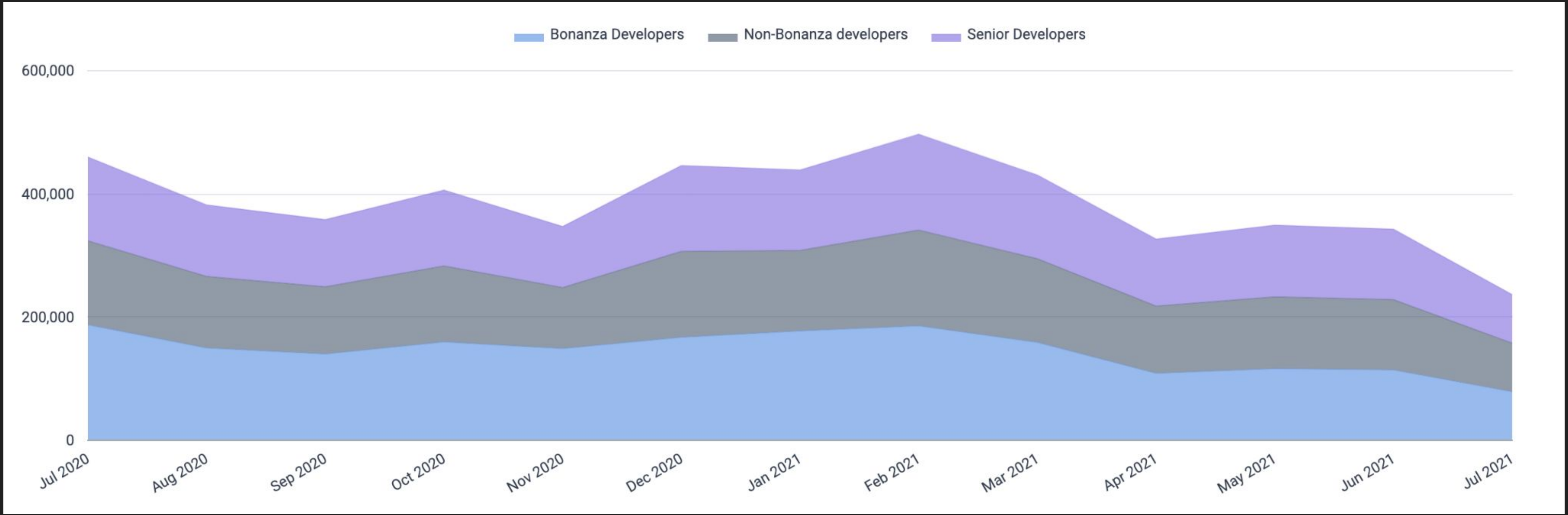
Encore reports for overachievers

Historic Stats: Long-Term Diff Delta



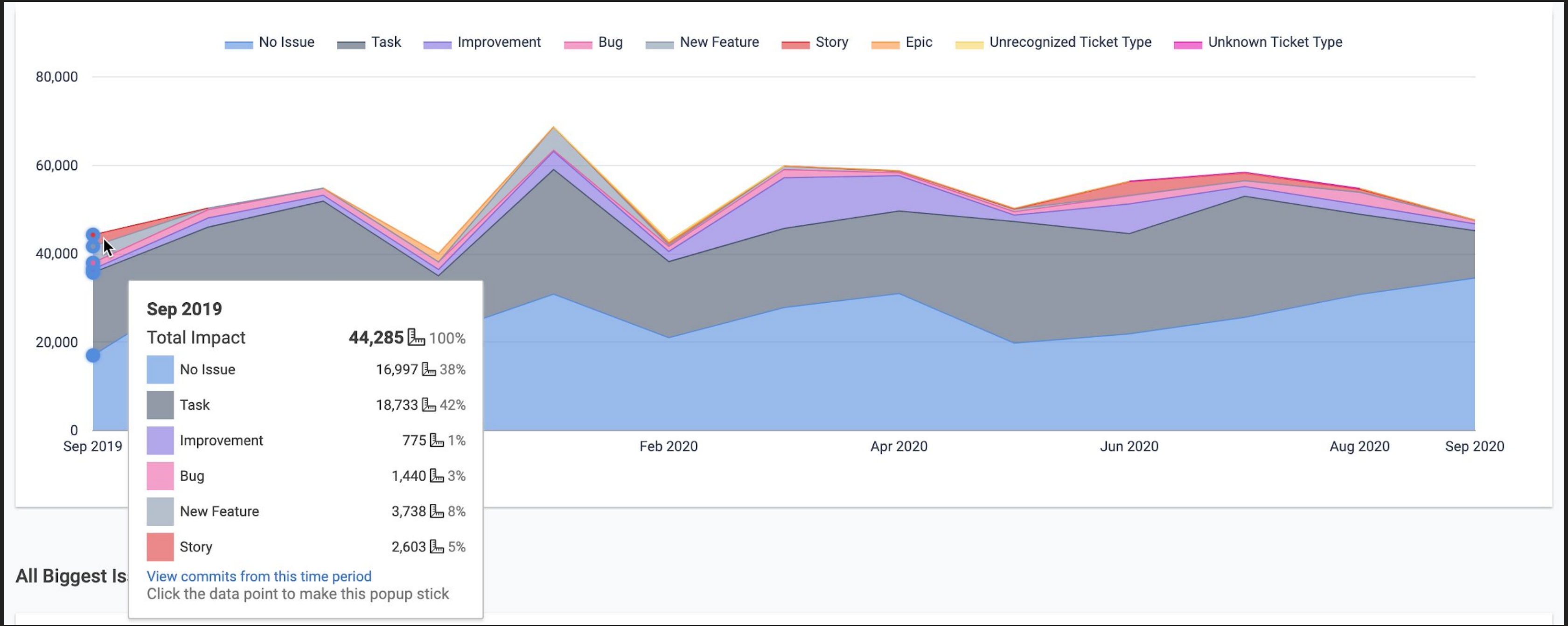
Historic stats show how temporal factors impact developer output

Historic Stats: Line Impact by Team



How are various teams at the company progressing on their tasks?

Issue Stats: How Much Work Comes From Jira?



Sep 2019

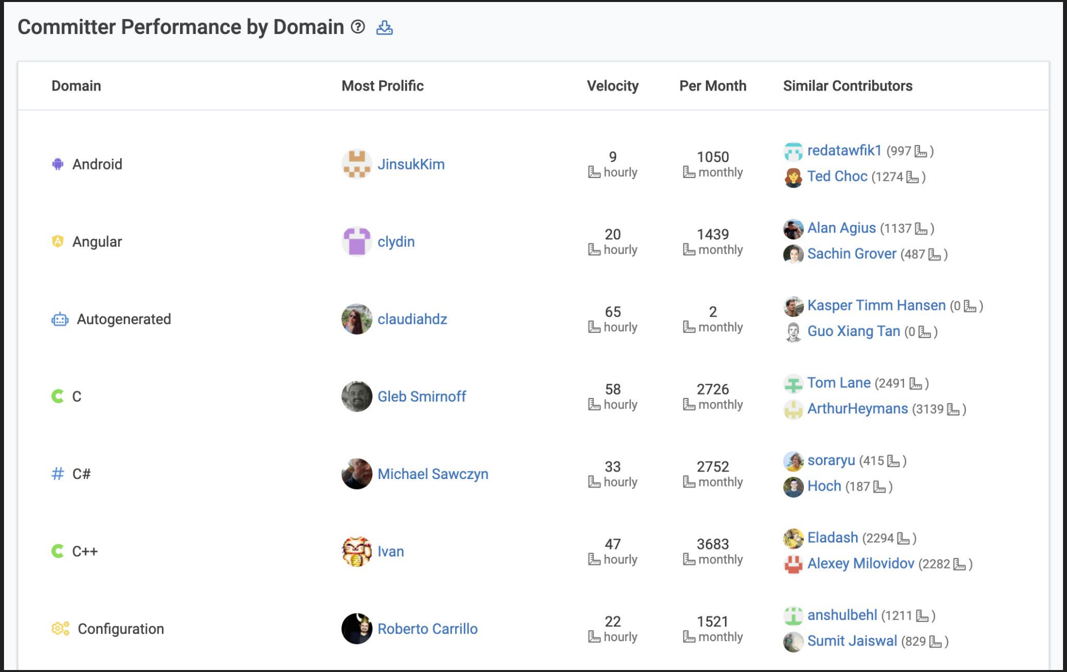
Total Impact	**44,285**	100%
No Issue	16,997	38%
Task	18,733	42%
Improvement	775	1%
Bug	1,440	3%
New Feature	3,738	8%
Story	2,603	5%

View commits from this time period

Click the data point to make this popup stick

Issue stats make clear when bugs increase, or when the team is working on undocumented work

Your Team Has Experts, So Who Are They?



Committer Performance by Domain

Domain	Most Prolific	Velocity	Per Month	Similar Contributors
Android	JinsukKim	9 hourly	1050 monthly	redatawfik1 (997) Ted Choc (1274)
Angular	clydin	20 hourly	1439 monthly	Alan Agius (1137) Sachin Grover (487)
Autogenerated	claudiahdz	65 hourly	2 monthly	Kasper Timm Hansen (0) Guo Xiang Tan (0)
C	Gleb Smirnoff	58 hourly	2726 monthly	Tom Lane (2491) ArthurHeymans (3139)
C#	Michael Sawczyn	33 hourly	2752 monthly	soraryu (415) Hoch (187)
C++	Ivan	47 hourly	3683 monthly	Eladash (2294) Alexey Milovidov (2282)
Configuration	Roberto Carrillo	22 hourly	1521 monthly	anshulbehl (1211) Sumit Jaiswal (829)

Actual developer data, aggregated across 20 of the largest open source repos over past 12 months

A customer favorite, our Domain Experts report uses git data to identify who in the company has the most experience writing every type of code across your projects.

These people can form the interview team if you need to hire more experts. They're also prime candidates if you need to mentor aspiring Junior Developers.