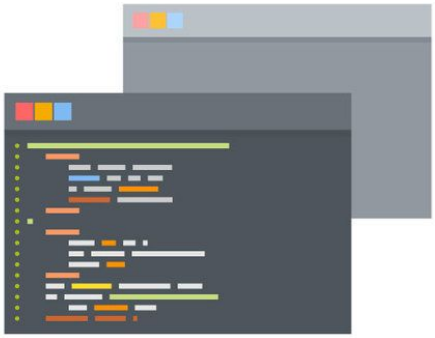
LosCat: A Tool for Source Code Monitoring

Anders Bruihler, Ari Conati,
Sebastian Kimberk, & David White

Background - Source Code

General idea of programming as someone sitting at a computer typing code.

Background - Source Code



Breaking down that idea, there are two main parts - the programmer, and the code.

Background - Source Code

If you are less familiar with programming and CS in general, you can basically think of code as a big Word document. It's not a perfect analogy, but close enough where it'll work for comparison.

Background - Source Code

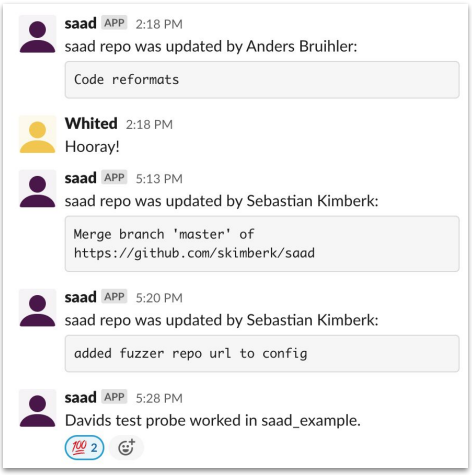
One important aspect of prorgramming is that you'll almost always be working not by yourself, but rather with a group of people. The group size will vary, but it still comes down to everyone working on the same code/Word document at the same time. There are tools (git, etc.) that help manage this, but this basic idea is still true.

Background - Source Code Monitoring

Coding - a big group project with people working on different parts

Natural desire to monitor the changes!

- Tests failing
- Inconsistent formatting
- etc.



If you think of coding as a big group project with lots of different people working on different parts of the same big Word document, you've got a reasonable grasp of how programming generally works today.

Given this group project dynamic, there is a natural desire to monitor changes that arises.

For example, you might want to be notified when tests fail, when someone adds inconsistent formatting, or in any number of other similar situations.

In terms of the Word document version of programming, you might want to be notified when someone changes the conclusion that you have put a lot of work into, when people are working on the document in general, or something else. If you've ever worked in a group on a document, you've probably had these sorts of thoughts in the past.

The idea behind source code monitoring is to automatically watch the changes that happen to code (i.e. whenever someone clicks the save icon in the group Word document), and be able to react to things that happen. This is what our tool enables people to do.

You can see a couple examples here:
On the right, you can see our Slack history. We set up LosCat to automatically send a message when someone makes a change with the name of the person and the

message that they described their change with.
In the middle, you see a notification that is sent when a new version of a program is released, and the notification includes the new version number as well as a list of people who contributed to the project since the last version. We'll get more in depth with how to do this specific example later.
On the bottom left, you see a screenshot saying that Ari has changed an important part of the code, and that further attention may be warranted. This message is only sent when this specific part of the code (Word document) has been changed.

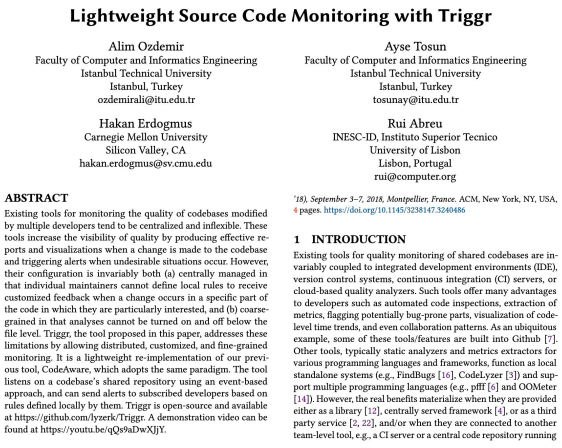
Inspiration - Triggr

"Lightweight source code monitoring with Triggr" - Automated Software Engineering 2018 Tool Demonstrations

Previous solutions: centralized, inflexible, coarse-grained

Triggr goals: distributed, flexible, fine-grained

How: "Probes" tracked in the repo, customizable by developers, targets file/class/method, triggers "actuators"



Lightweight Source Code Monitoring with Triggr

Alim Ozdemir
Faculty of Computer and Informatics Engineering
Istanbul Technical University
Istanbul, Turkey
ozdemirali@itu.edu.tr

Ayse Tosun
Faculty of Computer and Informatics Engineering
Istanbul Technical University
Istanbul, Turkey
tosunay@itu.edu.tr

Hakan Erdogmus
Carnegie Mellon University
Silicon Valley, CA
hakan.erdogmus@sv.cmu.edu

Rui Abreu
INESC-ID, Instituto Superior Tecnico
University of Lisbon
Lisbon, Portugal
rui@computer.org

ABSTRACT

Existing tools for monitoring the quality of codebases modified by multiple developers tend to be centralized and inflexible. These tools increase the visibility of quality by producing effective reports and visualizations when a change is made to the codebase and triggering alerts when undesirable situations occur. However, their configuration is invariably both (a) centrally managed in that individual maintainers cannot define local rules to receive customized feedback when a change occurs in a specific part of the code in which they are particularly interested, and (b) coarse-grained in that analyses cannot be turned on and off below the file level. Triggr, the tool proposed in this paper, addresses these limitations by allowing distributed, customized, and fine-grained monitoring. It is a lightweight re-implementation of our previous tool, CodeAware, which adopts the same paradigm. The tool listens on a codebase's shared repository using an event-based approach, and can send alerts to subscribed developers based on rules defined locally by them. Triggr is open-source and available at https://github.com/lyzerk/Triggr. A demonstration video can be found at https://youtu.be/qQs9aDwXJjY.

'18), September 3–7, 2018, Montpellier, France. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3238147.3240486

1 INTRODUCTION

Existing tools for quality monitoring of shared codebases are invariably coupled to integrated development environments (IDE), version control systems, continuous integration (CI) servers, or cloud-based quality analyzers. Such tools offer many advantages to developers such as automated code inspections, extraction of metrics, flagging potentially bug-prone parts, visualization of code-level time trends, and even collaboration patterns. As an ubiquitous example, some of these tools/features are built into Github [7]. Other tools, typically static analyzers and metrics extractors for various programming languages and frameworks, function as local standalone systems (e.g., FindBugs [16], CodeLyzer [3]) and support multiple programming languages (e.g., pfff [6] and OOMeter [14]). However, the real benefits materialize when they are provided either as a library [12], centrally served framework [4], or as a third party service [2, 22], and/or when they are connected to another team-level tool, e.g., a CI server or a central code repository running

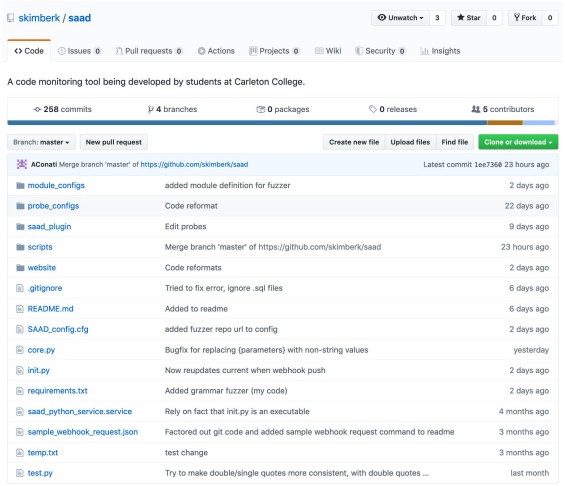
We were inspired by Triggr, which is a source code monitoring tool created and presented at the ASE 2018 conference, in the Tool Demonstrations section.

The authors found that previous solutions required developers to come to agreement on how the monitoring tool should be configured, with limited configuration options.

They wanted Trigger to allow individual developers to set up their own configurations without having an impact on others, while also enabling developers to create exactly what they want.

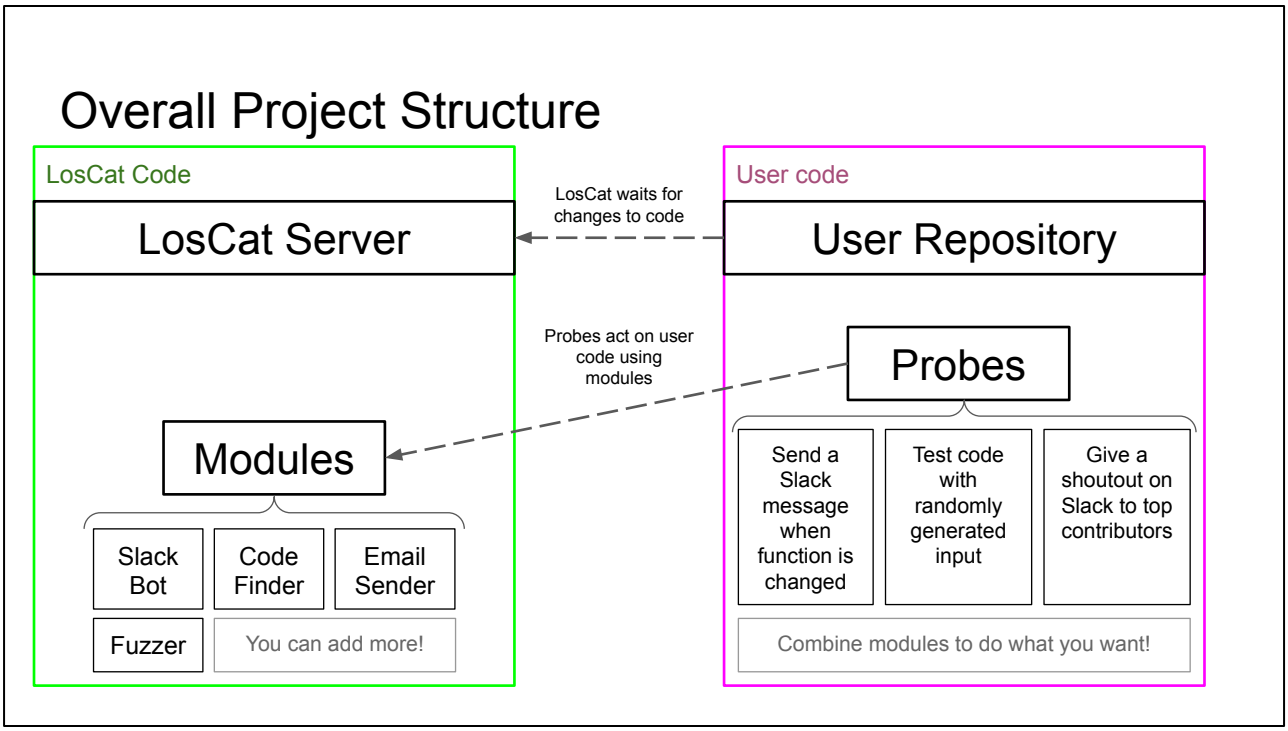
Goals/Improvements to Triggr

- Do it ourselves!
- In Python (instead of .NET Core)
- Support targeting Python methods, classes
  - As well as generic file changes
- Include:
  - Slack bot
  - Email
  - Fuzzer
  - etc.
- IDE plugin
- Extensibility



Benefit of doing in Python - much more popular and easier language in .NET Core, so benefit to those who use LosCat. Also we all had experience with Python.

A big goal of LosCat was to let users easily add their own functionality to accomplish whatever they want. We've created some default modules, but adding new stuff is very easy.



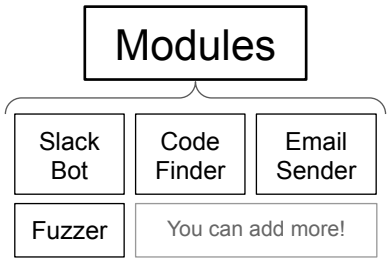
In this diagram the Loscat Code is our code and the User Repository is the project for which a user wants to implement source code monitoring. Modules are scripts which are useful for performing various code monitoring tasks. Users configure probes by supplying parameters for the modules they want to run on their code. These custom probe files are stored in the user repository. The LosCat server then monitors the User Repository, and whenever an update is pushed, LosCat automatically runs the configured probes on the user's code.

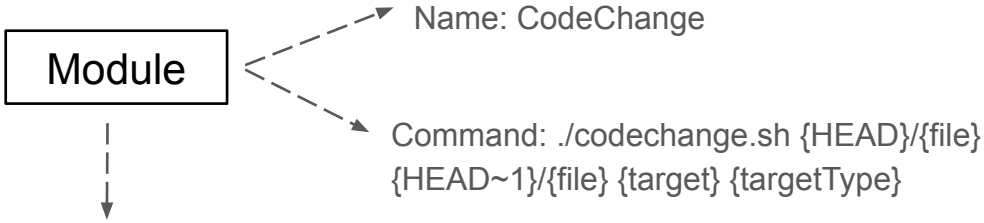
Example - Tracking Code Changes

- User wants to be notified when the function foo is changed by another developer
- Probe sends a slack message to the designated channel when foo is changed

Module structure

- Modules are scripts that might be run as part of a code monitoring task



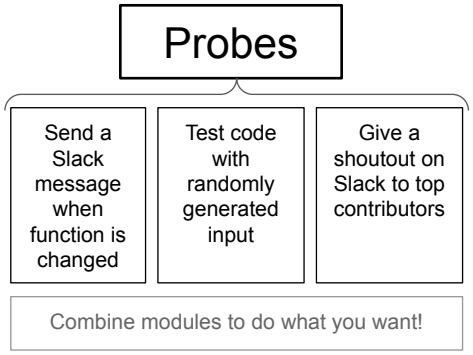


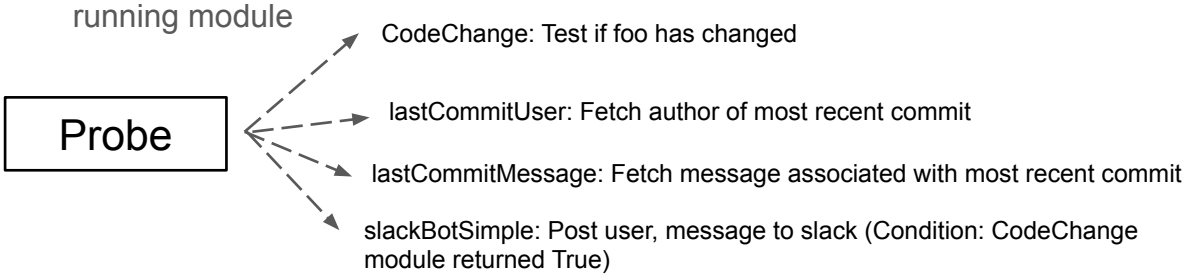
Return: True if the piece of code specified by target (file, function, class) has changed between git commits, False otherwise

Modules are stored as a JSON list, where each module consists of its name and command line command. Parameters are indicated within the command using '{}'.

Probe Structure

- Probes supply parameters for modules to perform a code monitoring task
- Chain modules to perform more sophisticated tasks
- Can include condition for running module





Probes contain a list of modules with parameters supplied. Each probe configuration file is a JSON list. Each script in a probe is run in order (may actually be run in parallel, but for the purposes of writing the files it is best to think of them as running top to bottom). Each module also has an optional condition indicating whether it should be run at all.

Example - Tracking Code Changes

- User wants to be notified when the function foo is changed by another developer
- Probe sends a slack message to the designated channel when foo is changed

```
[ {
  "name" : "simpleCodeChange",
  "type" : "codeChange",
  "config" : {
    "file" : "src/test.py",
    "targetType" : "function",
    "target" : "foo"
  }
}, {
  "name" : "getUser",
  "type" : "lastCommitUser",
  "config" : { }
}, {
  "name" : "getMessage",
  "type" : "lastCommitMessage",
  "config" : { }
}, {
  "name" : "slackMessage",
  "type" : "slackBotSimple",
  "config" : {
    "condition" : "{simpleCodeChange}",
    "channel" : "my-channel",
    "message" : "function foo updated by {getUser}: {getMessage}"
  }
} ]
```

Modules

Slack Bot

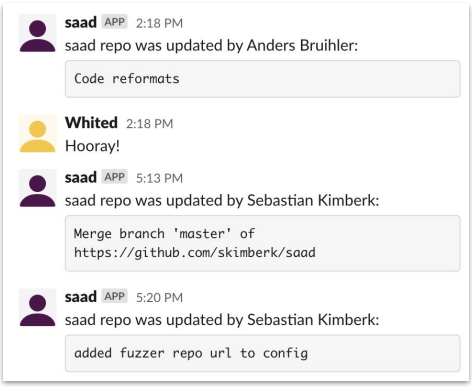
Python AST

Fuzzer

Custom modules

Modules: Slack Bot

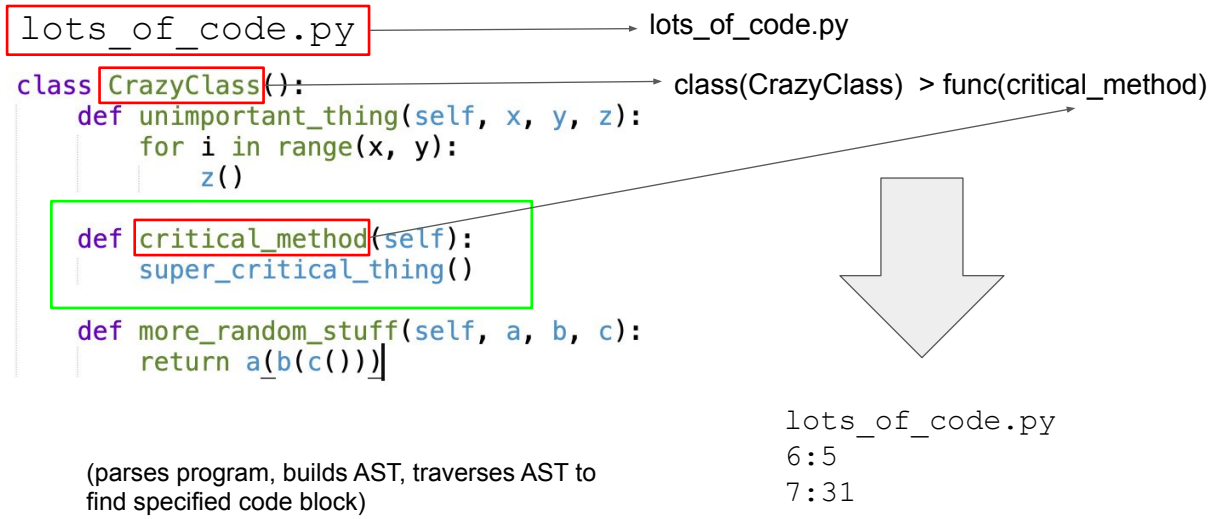
- Python program with two modes
  - `$ python3 slack_actuator.py --simple [API_TOKEN] [channel] [text]`
    - Basic text message
  - `$ python3 slack_actuator.py --blocks [API_TOKEN] [channel] [text] [blocks]`
    - Block Kit-formatted message
- Corresponding LosCat modules:
  - slackBotSimple
  - slackBotBlocks
  - Module code handles API_TOKEN
- Extensions?
  - This is a simple but useful bot
  - Potential for interactivity with Slack user



For the majority of the time that we were working on LosCat, we had it running and sending notifications to our Slack channel whenever someone made a change, as shown in the picture. We found this incredibly helpful in terms of awareness of work that was happening on the project.

The current version of the Slack bot is a quite basic implementation. There is a lot of potential to add more support, including sending messages to dynamically calculated people (instead of a hard-coded channel for each probe), and also to add interactivity with developers through Slack. For example, a message could present the developer with two buttons, and depending on the input, further actions could be taken by LosCat automatically.

Modules: Python Code Finder



What is an AST? Abstract syntax tree? Define the jargon. Make it make sense to a friend who would struggle to understand this.

Use case: You want to run a probe on a specific piece of code (for example, a specific function or class)
The Python AST module takes in a simple syntax to specify code location
class(TestClass): matches class(es) named TestClass
class(TestClass) func(simple_function): function anywhere inside the class
class(TestClass) > func(simple_function): function that's a direct descendant of the class
> func(simple_function): only global functions (direct children of root)
Analyzes Python AST in order to find code
Returns the start location and end location (the lines and columns)
(Example: codechange.sh tells you when AST changes, slide notes)

```
[
 {
        "name": "myMethodUpdated",
        "type": "codeChange",
        "config": {
        "file": "core.py",
        "target": "my_method",
        "targetType": "function"
        }
```

```
  },
  {
          "name": "user",
          "type": "lastCommitUser",
          "config": {
          }
  },
  {
          "type": "slackBotSimple",
          "config": {
          "condition": "myMethodUpdated",
          "channel": "[your DM ID]",
          "message": "{user} changed my method!  What are they doing!"
          }
  }
]
```

Video demonstration of a message being sent when critical_method is changed, and not when another message is changed.

Modules: Grammar Fuzzer

- What's a fuzzer?
- Grammar?
- Use case?

What is a fuzzer? What's a grammar? Define the jargon. Make it make sense to a friend who would struggle to understand this. ANTLR.

Use case: a program takes input in a specific form. We want to automatically test the program with a wide range of syntactically correct inputs.
Example: a Scheme interpreter
The probe takes a grammar definition (specified in ANTLR format) and a command, then it generates input from the grammar and passes it to the command via STDIN
All errors are recorded and output from the probe

Fuzzer Example: Grammar

```
grammar POSTFIX_CALC;

expr
   : NUMBER
   | expr ' ' expr ' ' OPERATOR;

OPERATOR : '+' | '-' | '*';


// Borrowed from JSON.g4 from antlr4-grammars/

NUMBER
   : '-'? INT ('.' [0-9] +)? EXP?
   ;


fragment INT
   : '0' | [1-9] [0-9]*
   ;

// no leading zeros

fragment EXP
   : [Ee] [+\-]? INT
   ;
```

An expression can be a number (i.e. 47 or -17.375)

Or, two expressions followed by an operator.

Examples:
1 2 +
1 2 + 3 *

Fuzzer Example: Grammar Output

0.8721

927E6 -0.9 -

9 -3.45e-0 -45.702686618 * -324 * 2.13 - 0E-0 * 602502.02 -0.7 * 19.10 * 488e+0 * - 0.398e-63 + *

-0 -8e50 9442 -3e56 * 0E-49 * - 8E82 -0.3e9 3 + 0E1 + 4.4 0.5184E0 0.15E0 -0 0.2e3 - 0.7e0 0.0E0 -0.33E0 8.80654e6 + 8E88 -9.09E+0 -0.3 - - * 0e-6 - * -5 * 0.7e-0 * 45.718 + * + * - + 0e83 + 0.3663e0 - * -0.4E0 0E157 * 4E9 - + * -0.2E4957 -1 + 0.7E+4 0.9 - - + - - -0E-59630 -0.33827 -85.9 13.4 -3.751 * 0.766 * 0 0.6e0 6 * + - 61.49078 -0 * -3.92e-0 -0 - 0.222 0.25e+6 - -67e0 + + - + -0 + * * - -0.7 - *

(generates graph representation of grammar, calculates minimum recursion depth to generate each node, randomly traverses graph while keeping track of depth)

Fuzzer Example: Probe

```
[
    {
        "name": "fuzzOutput",
        "type": "grammarFuzz",
        "config": {
            "grammarFile": "POSTFIX_CALC.g4",
            "entryRule": "expr",
            "executeFile": "postfix_calculator.py"
        }
    },
    {
        "type": "slackBotSimple",
        "config": {
            "channel": "monitoring-slack-test-public",
            "message": "{fuzzOutput}"
        }
    }
]
```

Specify grammar file

Specify grammar rule to generate

Specify what file to run with generated input

Send results to Slack!

Example running of a the fuzzer probe, including demonstration of how input producing errors is sent to Slack when there is a bug with the tested code.

Custom module/probe example walkthrough

Idea: On new version release, post a message to Slack announcing the release and thanking contributors to the release

This walkthrough will highlight different aspects of LosCat to show how it can easily be extended, in order to add specific desired functionality without much cost to the developer.

This message is automatically created and sent when version.txt is changed, and the contributors are calculated since the last git tag.

Custom walkthrough - probe overview

```
if [version.txt changed]:
    postToSlack(
        …[new version.txt]…
        …[list of contributors since [last git tag]]…
    )
```

[version.txt changed] - fileChange default module
postToSlack - slackBotBlocks default module
[new version.txt] - readFile default module
[last git tag] - we need to create this as a new module (lastTag)
[list of git contributors since [...]] - we need to create this as a new module (committersSince)

Custom walkthrough - modules

```
"lastTag": {
  "command": "cd {HEAD} && git describe --tags --abbrev=0"
},
"committersSince": {
  "command": "./scripts/probes/committers.sh {dir} {tag}"
}
```

lastTag - simple bash shell command, using {HEAD} to cd into the directory where the new version of the code is
committersSince - implemented in an external bash shell script, needs two parameters, {dir} for the directory of the repo and {tag} for when to calculate contributors since.

committersSince is a bash script, but easily could be created in Python or whatever the developer is familiar with.

Custom walkthrough - committersSince bash script

Bash script that lists contributors to a git repo since a given tag

```
#!/bin/bash

cd "$1" || exit

committers=$(git shortlog "$2"..HEAD -sn)

echo "$committers"
```

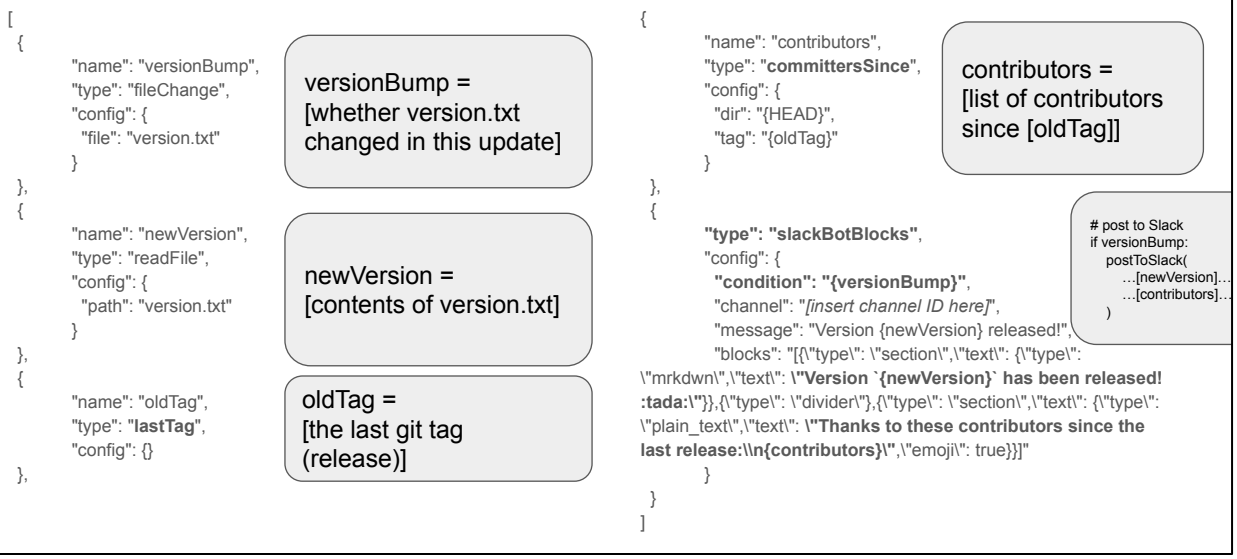
Output (commit count, name):

```
$ ./scripts/probes/committers.sh [directory] [tag]

      13    Alice

      12    Bob

      9     Eve
```

Actual script isn't really important, other than when given parameters it prints out the list of contributors. The output can be saved and used by another module.

(This could be accomplished without an external shell script, but this is used as an example.)

Custom walkthrough - probe



Here the different modules, including default ones, are combined to create the probe.

The name field is used to save the output of a module to use it if necessary later.

Note the condition field for slackBotBlocks - the message will only send if {versionBump} has changed - what we want!

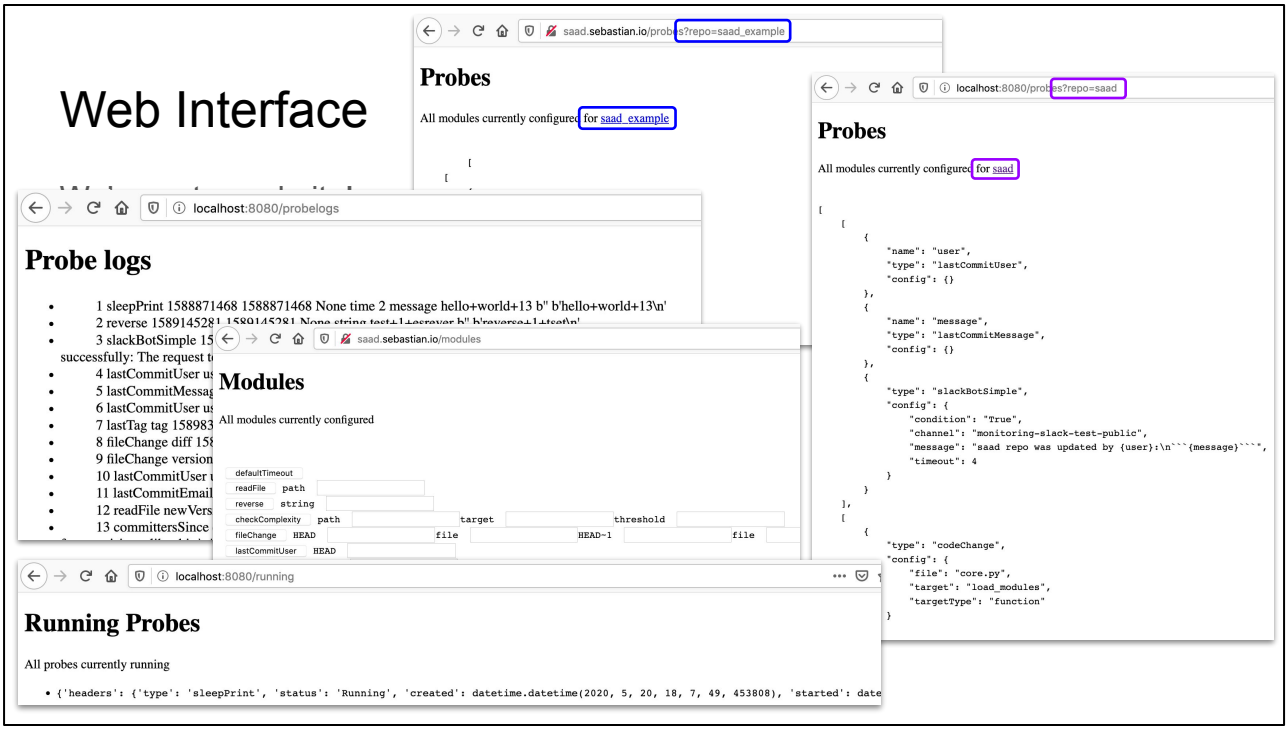
slackBotBlocks blocks field is complicated because it's a Block-Kit formatted message, which lets us make it look a bit better. Important parts (in bold) are that we include {newVersion} and {contributors}.

Custom walkthrough - result

It works! Yay!

This hopefully has been a good example of how a developer with a specific desire for code monitoring and automation on changes can easily create what they want in LosCat. We've shown a few different aspects of the process, including figuring out what modules are necessary, simple module creation, more advanced module creation, and how to put together the final probe.

If you've ever worked on a group project it is very possible that you had a moment where you wished you could be notified when something specific changed. If you've worked with code, this is even more likely. We've tried to have LosCat let you bring these into reality easily.



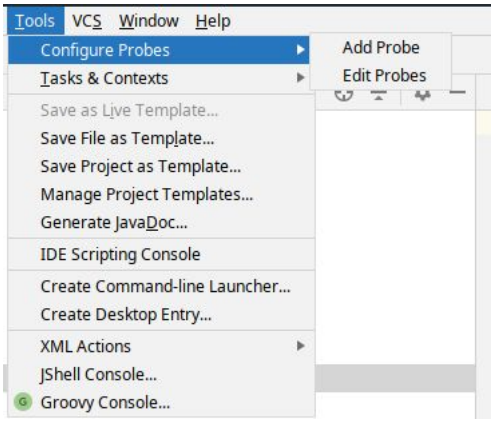
We have a website! It doesn't look great, but it is functional, which is the important part for now.

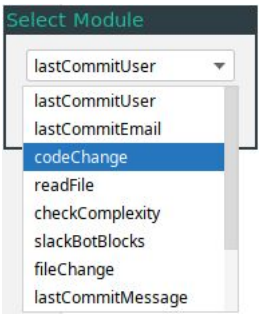
/probes&repo=?{repo_name} shows probes for the given repo that LosCat is watching (we support multiple repos per instance of LosCat)
/probelogs shows the logs of all run probes. Helpful for checking what/why a probe output and debugging.
/modules shows all the configured modules, and by filling in the fields, you can trigger a module.
/running shows the currently running probes. (Potential for a kill/restart/etc. button here, but we didn't get to that)

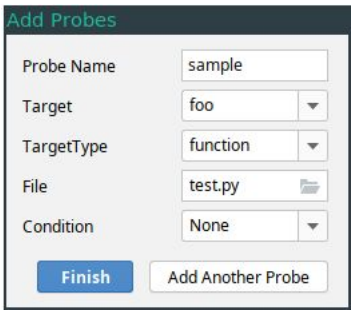
IDE Plugin Interface

- Plugin for JetBrains IDEs (IntelliJ, PyCharm, etc.) which allows for easily adding probes to a project
- Automatically generate probe JSON files

IDE Plugin Interface







Internals: Running A Probe

```
"codeChange": {
  "command": "./scripts/probes/code_change.sh
},
```

```
{
"name" : "simpleCodeChange",
"type" : "codeChange",
"config" : {
  "file" : "
  "targetType" : "
  "target" : "
}
```

2

1

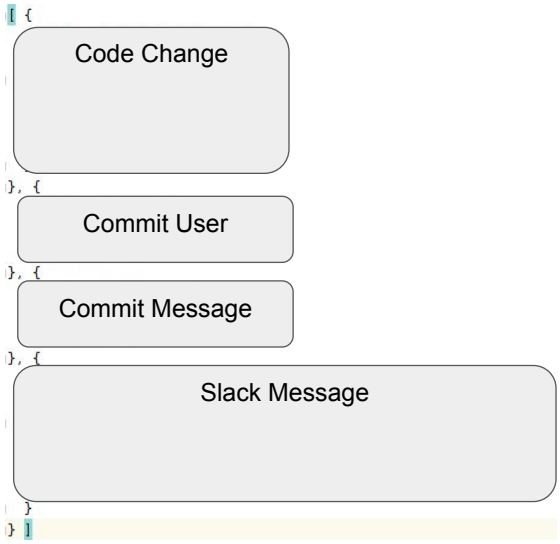
Extra Bindings

{HEAD}:

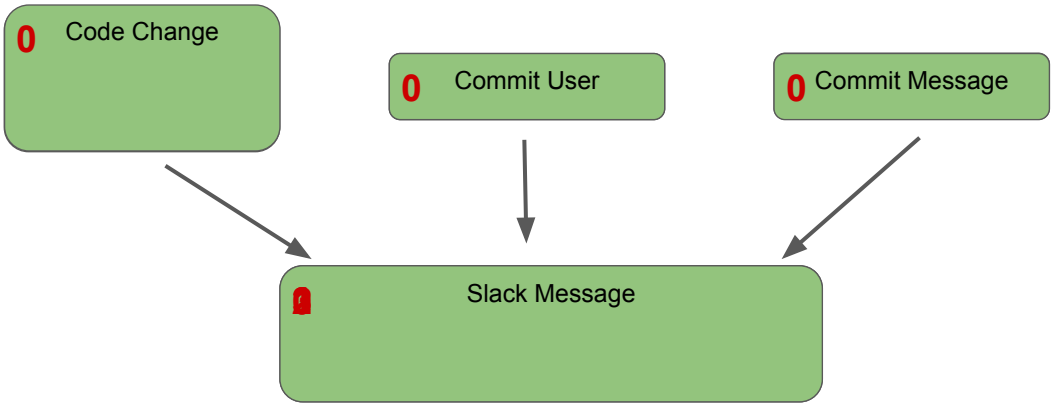
{HEAD~1}:

{simpleCodeChange}: “True”

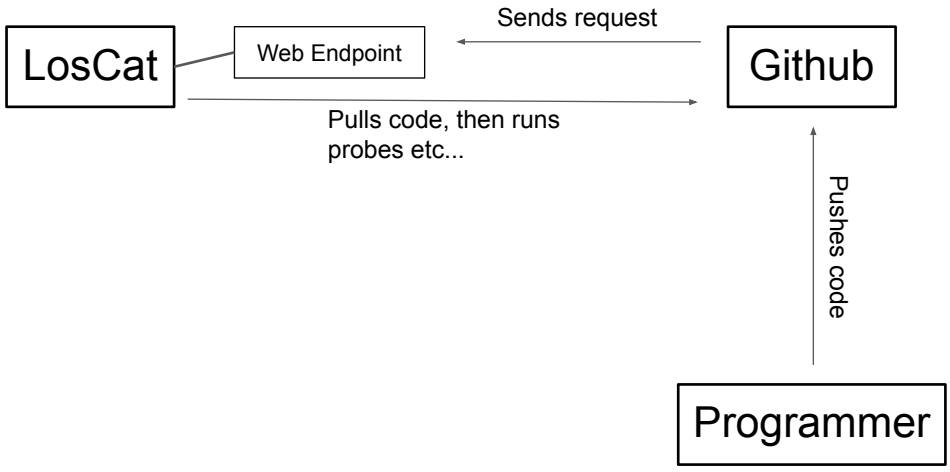
Internals: Running Multiple Probes



Internals: Running Multiple Probes



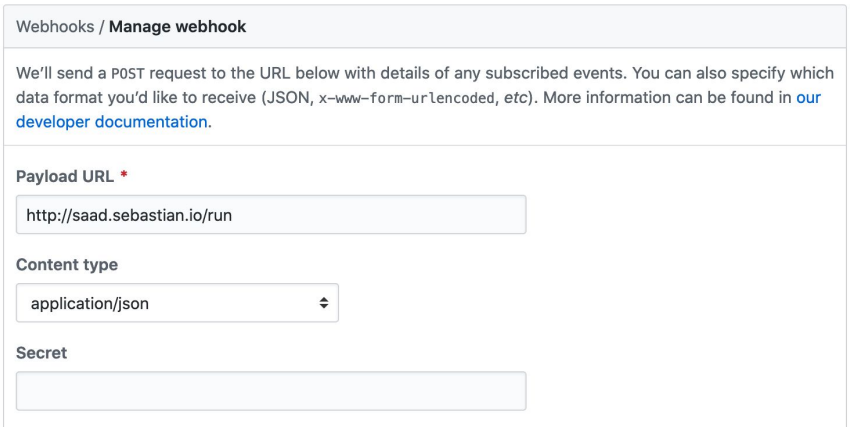
Internals: Github Webhooks



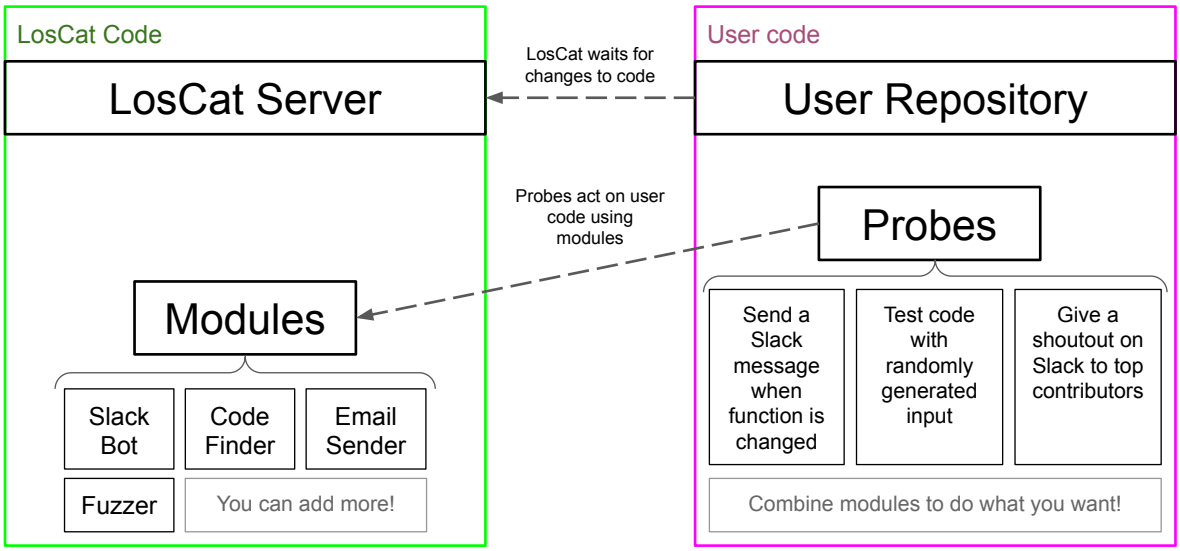
How does LosCat known when to run probes on your code?
Github has webhooks: you can easily setup your repo so that Github makes a POST request to a given URL whenever the repo is pushed to
When LosCat receives this request, it clones the repo, runs the probes, etc!

Internals: Github Webhooks

Super easy to set up!



Repository Management



Repository Management

LosCat Code

LosCat Server

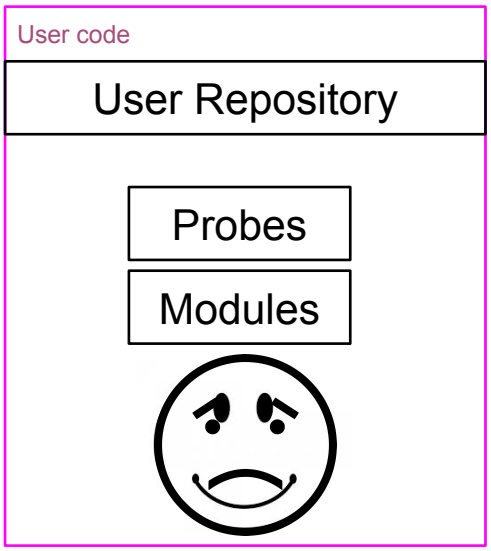
Modules

User code

User Repository

Probes

Repository Management

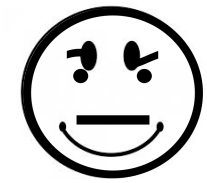


Repository Management

LosCat Code

LosCat Server

Modules

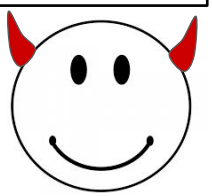


User code

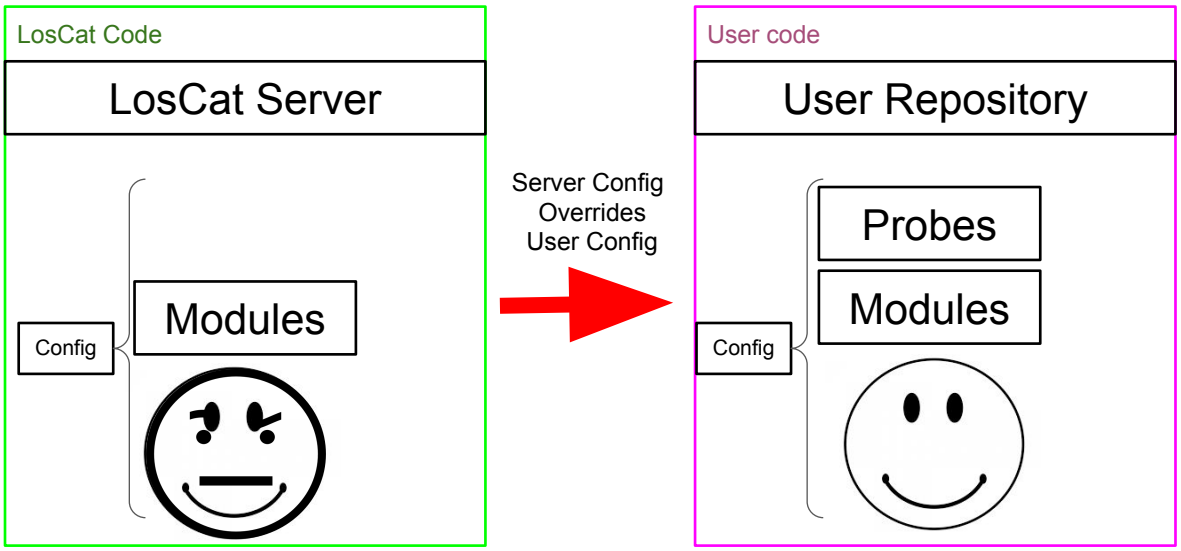
User Repository

Probes

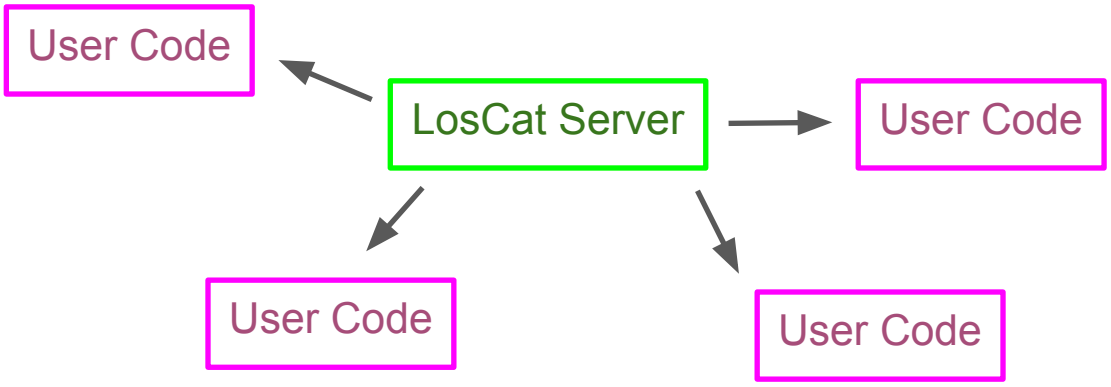
Modules



Repository Management

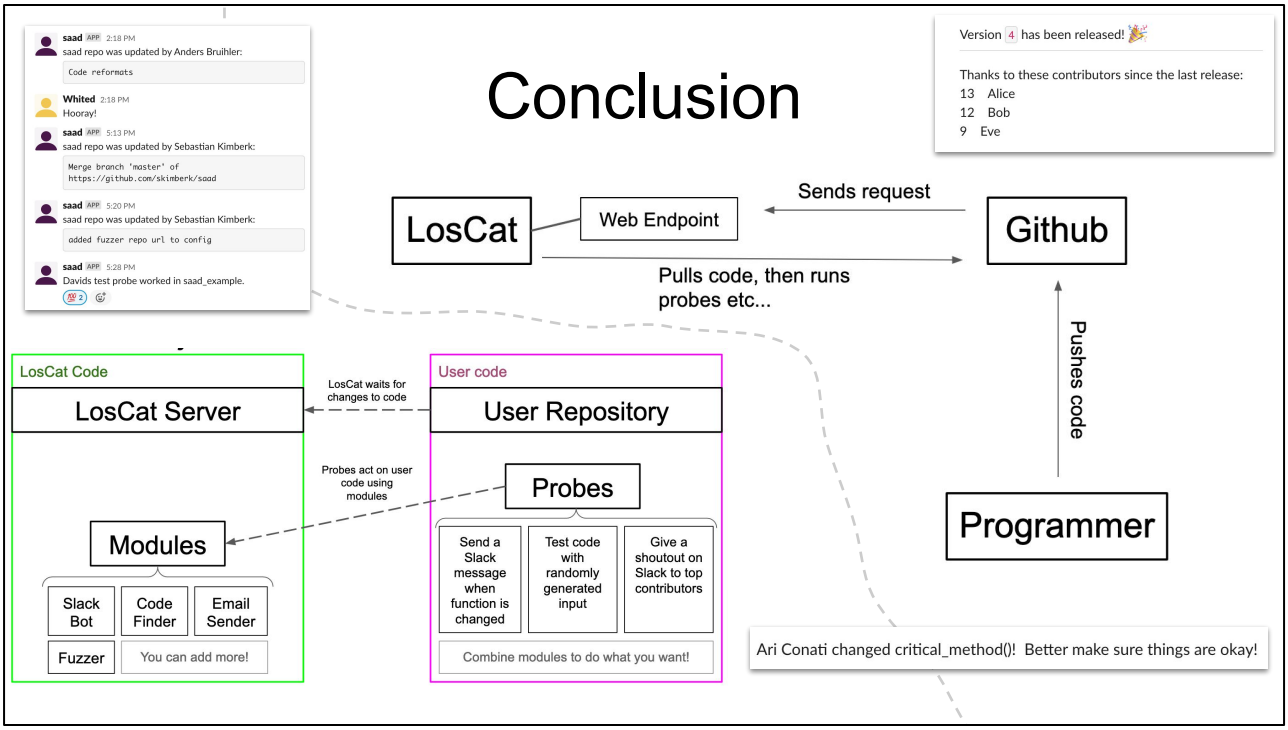


Repository Management



Internals: Server Configuration/Deployment

- Running on an AWS EC2 instance (which runs an Amazon variant of Redhat)
- Set up as a systemd service
  - Automatically restarts on crash/server restart
  - Service has a dedicated user with restricted permissions (can only modify the LosCat directory)
  - Runs on port 8080 (so as to not need sudo) and has forwarding from port 80 to 8080
- Python service listens for HTTP requests at a specific address. When hit, it pulls new code from Github, installs dependencies using pip, and it replaces the current process with the new code
  - Github is setup to hit the address whenever our repo is updated
  - Deployment is completely automated


saad APP 2:18 PM
saad repo was updated by Anders Bruihler:
Code reformats
Whited 2:18 PM
Hooray!
saad APP 5:13 PM
saad repo was updated by Sebastian Kimberk:
Merge branch 'master' of https://github.com/skimberk/saad
saad APP 5:20 PM
saad repo was updated by Sebastian Kimberk:
added fuzzer repo url to config
saad APP 5:28 PM
Davids test probe worked in saad_example.
Conclusion
Version 4 has been released!
Thanks to these contributors since the last release:
13 Alice
12 Bob
9 Eve
LosCat
Web Endpoint
Sends request
Github
Pulls code, then runs probes etc...
Pushes code
LosCat Code
LosCat Server
LosCat waits for changes to code
User code
User Repository
Probes act on user code using modules
Probes
Modules
Slack Bot
Code Finder
Email Sender
Fuzzer
You can add more!
Send a Slack message when function is changed
Test code with randomly generated input
Give a shoutout on Slack to top contributors
Combine modules to do what you want!
Programmer
Ari Conati changed critical_method()! Better make sure things are okay!

Thanks

- Dave Musicant
- The Carleton College CS Department
- Zoom, Slack, & Github
- Friends and Family
- You

Dave Musicant - our Comps advisor
Zoom, Slack, & Github - for allowing us to work together remotely

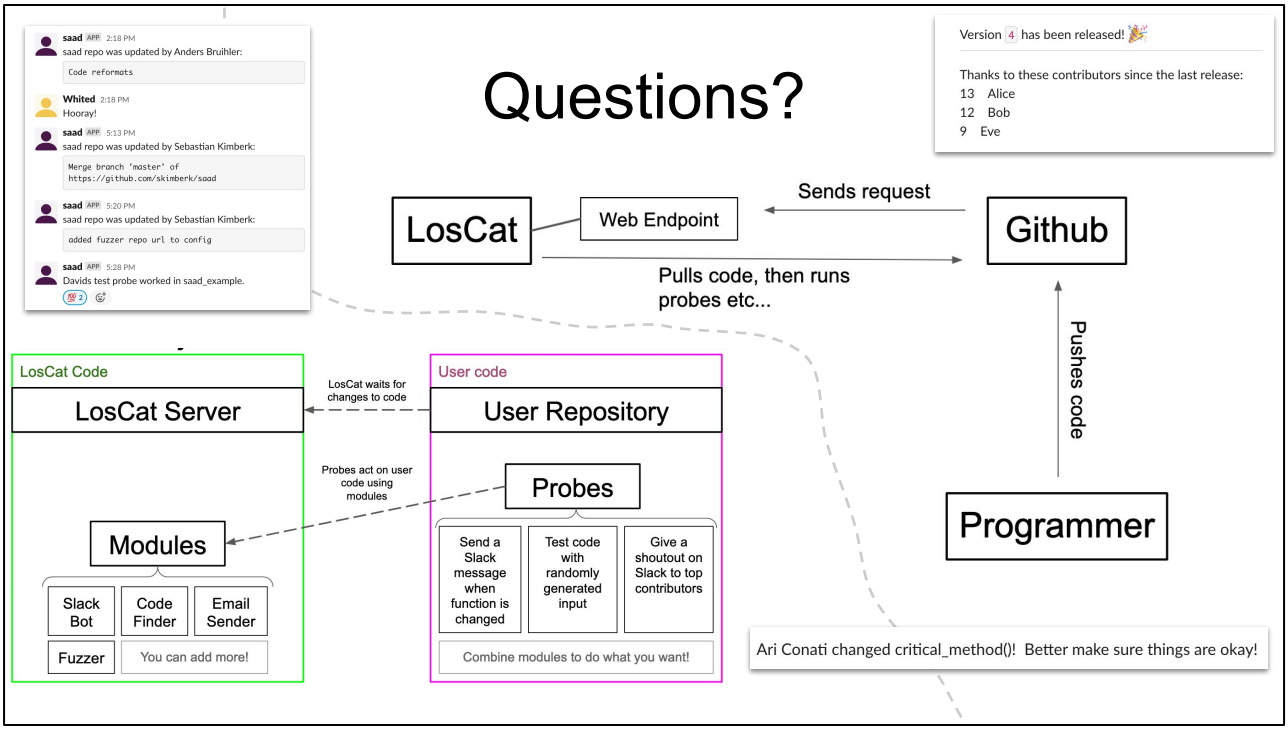
References

- “Lightweight source code monitoring with Triggr,” Automated Software Engineering 2018
  - <https://dl.acm.org/doi/10.1145/3238147.3240486>
- “CodeAware: sensor-based fine-grained monitoring and management of software artifacts,” International Conference on Software Engineering 2015
  - <https://dl.acm.org/doi/10.5555/2819009.2819099>
- “Grammarinator: a grammar-based open source fuzzer," A-TEST 2018
  - <https://dl.acm.org/doi/10.1145/3278186.3278193>

Triggr - inspiration paper
CodeAware - previous version of Trigger created by the authors

Images

Fuzzer Cat: <https://icatcare.org/advice/helping-your-new-cat-or-kitten-settle-in/>
Word icon: <https://commons.wikimedia.org/wiki/File:Microsoft_Office_Word_(2018%E2%80%93present).svg>
Profile: <https://publicdomainvectors.org/en/free-clipart/Anonymous-person/59504.html>
Code windows: <https://publicdomainvectors.org/en/free-clipart/Programming-language/70504.html>
Programmer + computer: <https://publicdomainvectors.org/en/free-clipart/Computer-programmer-from-back/80570.html>
Business people: <https://publicdomainvectors.org/en/free-clipart/Professional-people-silhouette/81904.html>
Smiley Face: <https://www.publicdomainpictures.net/pictures/130000/velka/clip-art-smiley-face.jpg>
Suspicious Face: <https://freesvg.org/img/Unknown-Smiley-Face.png>
Sad Face: <https://www.goodfreephotos.com/vector-images/unhappy-face-vector-clipart.png.php>


saad APP 2:18 PM
saad repo was updated by Anders Bruihler:
Code reformats
Whited 2:18 PM
Hooray!
saad APP 5:13 PM
saad repo was updated by Sebastian Kimberk:
Merge branch 'master' of https://github.com/skimberk/saad
saad APP 5:20 PM
saad repo was updated by Sebastian Kimberk:
added fuzzer repo url to config
saad APP 5:28 PM
Davids test probe worked in saad_example.
Questions?
Version 4 has been released!
Thanks to these contributors since the last release:
13 Alice
12 Bob
9 Eve
LosCat
Web Endpoint
Sends request
Github
Pulls code, then runs probes etc...
Pushes code
Programmer
LosCat Code
LosCat Server
LosCat waits for changes to code
User code
User Repository
Probes act on user code using modules
Probes
Modules
Slack Bot
Code Finder
Email Sender
Fuzzer
You can add more!
Send a Slack message when function is changed
Test code with randomly generated input
Give a shoutout on Slack to top contributors
Combine modules to do what you want!
Ari Conati changed critical_method()! Better make sure things are okay!