Good morning everyone. My name is Bryan Davis and my pronouns are he/him. I am a Principle Software Enginer working for the Wikimedia Foundation as a member of the Technical Engagment team and the manager of the Cloud Services subteam. I am excited to be able to talk to y'all today about tools created by the members of the Wikimedia technical community and how we can all help make them better.

Don't let your brother-in-law's niece be the only person who keeps your community running

Tools are a vital resource for on-wiki content creation and curation activities

This is a thesis that I came up with based on my personal experience in helping folks use Toolforge and conversations I've had with Wikipedians about how they do the things that the do on wiki. I used it as the core argument to change my job at the Wikimedia Foundation in 2016. That was when I started working with and for the Wikimedia technical contributor community full time. My passion is helping folks think about ways to make it easier to build and maintain tools. My goal is more tools that are better maintained with the presumption that that will also make things better, faster, easier for on-wiki communities.

Some of you are going to say [citation needed]

37.02%

Percentage of total edits made to Wikimedia wikis originating from tools and services hosted in Wikimedia Cloud Services, April-June 2019

I want to start with some fun numbers that I think help show that tools and bots are not just little toys that someone has made.

Over **55.4 Million** edits (**37.02%** of all edits) to wiki pages and wikidata entities

- **63%** of edits on www.wikidata.org **46.9M**
- **13%** of edits on commons.wikimedia.org
- **7.5%** of edits on en.wikipedia.org

This data was obtained by querying the checkuser log tables on the Analytics replicas.

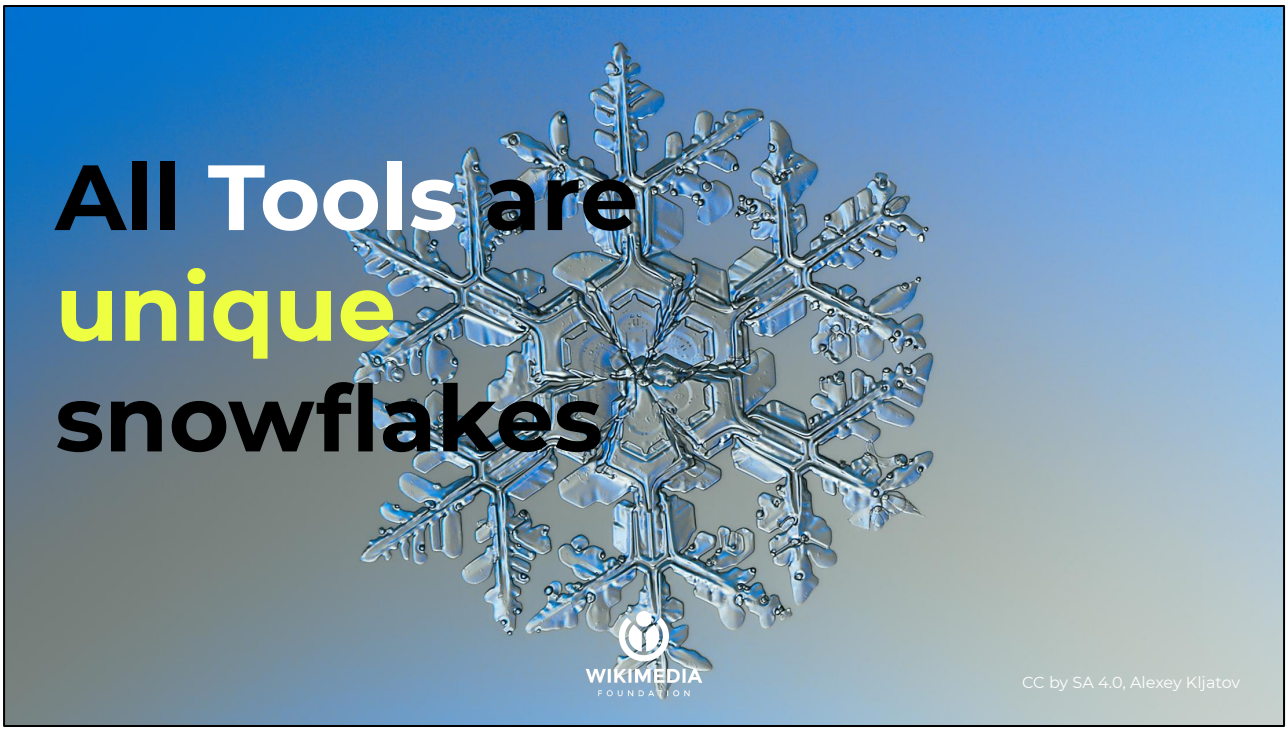
Time-to-revert by ClueBot NG's status

CLUEBOT NG STATUS	TIME-TO-REVERT, GEOMETRIC MEAN	MEDIAN
Up	941 seconds (15.7 minutes)	744 seconds (12.4 minutes)
Down	1674 seconds (27.9 minutes)	1286 seconds (21.4 minutes)

When the Levee Breaks: Without Bots, What Happens to Wikipedia's Quality Control Processes?
R. Stuart Geiger & Aaron Halfaker. (2013). WikiSym.

Aaron Halfaker and Stuart Geiger wrote a paper about the impact of just one bot in the first half of 2011. They found that it took roughly twice as long for garbage to get removed from the wikis when ClueBot NG wasn't running.

One study from data that's 8 years old now doesn't prove my thesis, but hey at least I have one credible source to point to. There are tools that help with detecting bad edits, tools that help with making proper citations, tools that help you find pages that are in need of various kinds of attention, tools that help stewards and patrollers figure out things about the edit habits of others. All of you in this room probably can tell me about a tool or two or three that you use regularly. Maybe you can even tell me about a tool that you bother to go on irc to whine about when its down.



A typical bot or tool project begins life as a way for a motivated Wikimedia community member to make some on-wiki task easier (or possible!). These individuals are "scratching their own itch" in the best tradition of open source development. Many of these projects have a short lifecycle due to factors such as loss of interest by the maintainer, insurmountable technical hurdles, or discovery of a better means to solve the original problem. A few however become popular and tightly integrated in the workflows of one or more on-wiki communities.

There is a wide range of experience and practices among the Toolforge developer community. Some tools are developed by professional software engineers with years of real world experience in designing and building highly reliable and maintainable software. Other tools are built by people who are just learning to write code by following online tutorials. Some Tool maintainers have years of experience as contributors to the Wikimedia projects and others are just discovering the Wikimedia world. Some tools are built with 100% from scratch code and others use many third-party frameworks and libraries. Some start with a group of like minded developers and some are solo works that have never been discussed with others. Tools are built using both well known and esoteric programming languages.

No level of experience, programming language, or process is intrinsically better or worse than another. The differences emerge over time. In my opinion, the best tools are the ones that end up fulfilling a need for an on-wiki community and have maintainers who remain responsive to requests from their users.

Some ways that things can go wrong

PD, Studio Lewy and Sons

I have two real world examples of tools that have serious issues that could have been avoided. I don't want you to leave today thinking that the developers of these tools are bad people or that they have failed the movement. These examples are presented as a retrospective to illustrate my broader points. We are not here to point fingers or place blame; we are here to learn what not to do next time. In that spirit, I'm going to try not to "name and shame" the tools involved directly. If you really want to know which exact tools I'm talking about you can dig around on Phabricator.

A phabricator bug is filed about a tool that is often down. Nothing too new there, except this particular tool is linked to in templates on many wikis. And these templates are used in quite a few pages: something like 20k direct transclusions on enwiki and 120k on dewiki.

Toolforge admins are aware of this tool and its stability issues. Yuvi (who is awesome by the way) has migrated it from the older Ubuntu Precise job grid hosts to the newer Ubuntu Trusty hosts and given it almost double the memory that a tool is granted by default to try and make it more stable. A user takes on monitoring as a pet project and updates the ticket regularly when the tool is down. Yuvi and Valhalla do a lot of restarts, but nobody ever seems to hear from the maintainer.

The tool is actually doing worse things than just being intermittently down however. It causes a memory leak in its webserver that begins to affect other tools on the job grid. Massive amounts of memory are being consumed and are only freed by stopping and starting the tool's webservice. Valhalla and Yuvi continue to investigate the issue, but they really need some support from the tool's maintainer.

After repeated pings on Phabricator and wiki talk pages, the maintainer shows up and explains that they have lost interest in this particular tool. They provide a link to the source code but decline to choose a software license. They instead state "you can do what you will with it". I tried a couple of times to get them to change their mind about this point, but thus far it has not happened. That's pretty much the end of the story. An unlicensed, unmaintained tool is a dead tool.

This next example shows how multiple small issues can compound over time. I watched this particular project go from needing a small update to being forced to shutdown entirely. Many people tried to help along the way, but ultimately some small omissions by the original tool author and external forces created a perfect storm that killed the tool.

The tool itself was a collection of cron jobs that had been approved to do many different tasks for a large Wikipedia project. I don't know the full history here, but I imagine that it followed similar patterns I have seen elsewhere. The author wrote a script to do some task that was needed on wiki. When that task was taken care of and things were working well someone pointed out another task that could use attention from a bot. Eventually this tool grew to have control over a large number of related curation tasks and made hundreds of useful edits on any given day. Everything was fine for days, weeks, months, maybe even years.

* July 2015: The bot is on the list of Action API consumers that were still using HTTP after the global switch to HTTPS. This was possible due to a loophole that had been left open in the server configuration for POST traffic. The tool maintainer responded that they would need Java 1.8 in order to fix their software. The maintainer is pointed to a Declined Phabricator task for the Java 1.8 upgrade.

* August 2015: Maintainer responds on the HTTP deprecation tracking task with a refusal to change the bot's coding to accomidate Java 1.7 due to the potential time investment.

* December 2015: Another user opens a new bug asking for Java 1.8. This is investigated and again found to be a problematic upgrade for Toolforge. When the upgrade bug is closed as declined, we opened a new bug specifically to look for a solution that will work for the tool. The maintainer again asserts that the fix would be easy if only we would provide the software upgrade that has now been rejected twice.

* May 2016: The solo maintainer has recently posted on another task that they do not have the ability to work on anything wiki related for at least a few weeks. Since we knew the maintainer was going to be AFK for awhile I used my Toolforge admin powers to look into how the tool was put together. After looking at the tool's scripts and configuration I found more bad news: there is no source code on the Toolforge server, only compiled jar files. This greatly limits what anyone can do to try and fix things.

* June 2016: We decided to try and make a special HTTP-to-HTTPS transparent proxy just for this tool. After the proxy was up we still had no response from the maintainer to help with testing it and time is running out. I decided that I would try to play the hero and make the fixes myself. I edited the TWENTY EIGHT job startup scripts to pass the correct arguments to the Java runtime and crossed my fingers that this would be all that was needed.

Sadly it was not. The particular libraries used by the tool didn't work with the standard Java HTTP proxy configuration values. I did further investigation and determined that there would be no way to fix the problem without changing the source code. Source I did not have access to because it was not present on the Toolforge server and not published by the tool maintainer.

The maintainer suddenly appeared on Phabricator right around this time and yet again asked for more special treatment for their tool. This is the same request for a newer version of Java that had been examined and rejected twice previously with quite detailed analysis as to why it was not a reasonable solution for Toolforge. They claim that the bot's source is licensed under the MPL, but no link to source code is provided.

On June 20th, 329 days after the first phabricator contact trying to warn of the issue, I shut down the cron jobs for the tool because the requests were all erroring out. One absentee maintainer, no source code, no license, procrastination, and demands for special treatment had killed the project.

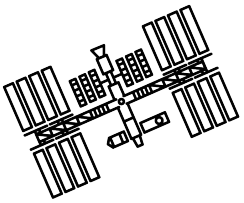
<read quote from slide>

The two example tools I've talked about have some common failings that are sadly widespread in Toolforge.

When you find a tool that helps you get a task done easier or better you want it to be there the next time you need it. This is especially important if the tool has become "critical" for some workflow that is needed to keep your wiki healthy. You, the users of tools, can create norms and ask that your tools adopt them. Developers generally will be excited to learn that others are using their tools and will be interested in making their users happy. Our movement is about sharing and collaboration on the technical side as much as it is on the content side.

Open Source practices

- ➔ Pick a license
- ➔ Publish the code
- ➔ Have multiple maintainers
- ➔ Write some documentation
- ➔ Participate in the Community



WIKIMANIA STOCKHOLM

So what kinds of norms should you create? I would suggest ones that are relatively easy to comply with and make sense to preserve the value of the tools for years to come. We can look to the larger ecosystem of Free and Libre Open Source Software projects for some basics to get started.

Pick a license
Neither tool had an explict license. Without a license you are implicitly claiming copyright without providing an explanation of the rights you are willing to grant to others who wish to use or modify your software. This means that you retain all rights to your source code and that nobody else may reproduce, distribute, or create derivative works until standard copyright lapses. In the US today that means until 70 years after your death. (That ridiculous duration will continue to grow too as long as Disney wants to protect Mickey Mouse.) This is counter to the general principles of the Wikimedia movement and a violation of the Toolforge terms of use.

Publish the code
There was no documented source code repository for either tool. The first tool is written in an interpreted language, so the current source is recoverable, but doing so requires access to the tool account itself. There are several gratis source code hosting options that tool developers can use. We also have a few libre options available thanks to the Wikimedia Foundation. The new tools admin console (<https://toolsadmin.wikimedia.org/>) makes creating a git repo for a tool's source code a one click task.

Have multiple maintainers
Having multiple maintainers is very helpful. Nobody has time to make sure that their tool is up and running 24/7/365 on their own. Having a few people who are familiar with at least starting and stopping a tool goes a long way towards improving uptime. It also creates a path for the original maintianer to transition out of a project when they eventually lose interest.

Write some documentation
Bug #1 (Phab T2001) is a problem for all software projects. A tools ususally doesn't need a lot of documentation but having a page on Wikitech, Meta, or another wiki that explains how to start and stop the tool and a bit about how to troubleshoot common problems goes a long way. We have the Tool namespace on Wikitech specifically for this purpose, but like I said put it wherever you'd like.

Participate in the Community
<next slide>

Technical Community

- Join our mailing lists:
  - cloud
  - cloud-announce
- Visit #wikimedia-cloud
- Use wikitech
- Use Phabricator

We know it takes a lot of people who are interested and skilled in a lot of different ways to create and maintain a wiki. The software that tool maintainers write is really no different. By acting as a community and supporting each other we can build better software. Asking questions of strangers can be scarry, but asking questions of your neighbors and friends is easier. Tool maintainers can and should get and give help to each other in friendly ways through a variety of channels.

THANK YOU!

Credits

- Snowflake macro photography 1.jpg By Alexey Kljatov, CC BY-SA 4.0
- Train wreck at Montparnasse 1895.jpg credited to Studio Lévy and Sons, Public Domain
- Stop to notice the little things (7360780594).jpg By Kenny Louie, CC BY 2.0
- Hurricane Frances from the ISS - 10AM. EDT AUG 27 2004.jpg By Mike Fincke, Public Domain
- Wmf logo vert K.svg By Logo and trademark of the Wikimedia foundation, designed by Wikipedia user Neolux, (SVG version created by DarkEvil, revised by Philip Ronan and optimized by Zscout370 and Artem Karimov) Later revised by Wikimedia Foundation, CC BY-SA 3.0
- Wikimania2019 logo.svg and related graphics By BFlores (WMF), CC BY-SA 4.0
- Various icons By Outstandy - Graphic designs for Wikipedia 15 from Mule Design, CC0

Copyright © 2019, Bryan Davis and the Wikimedia Foundation.
This work is licensed under a Creative Commons Attribution-Share Alike 4.0 International license.