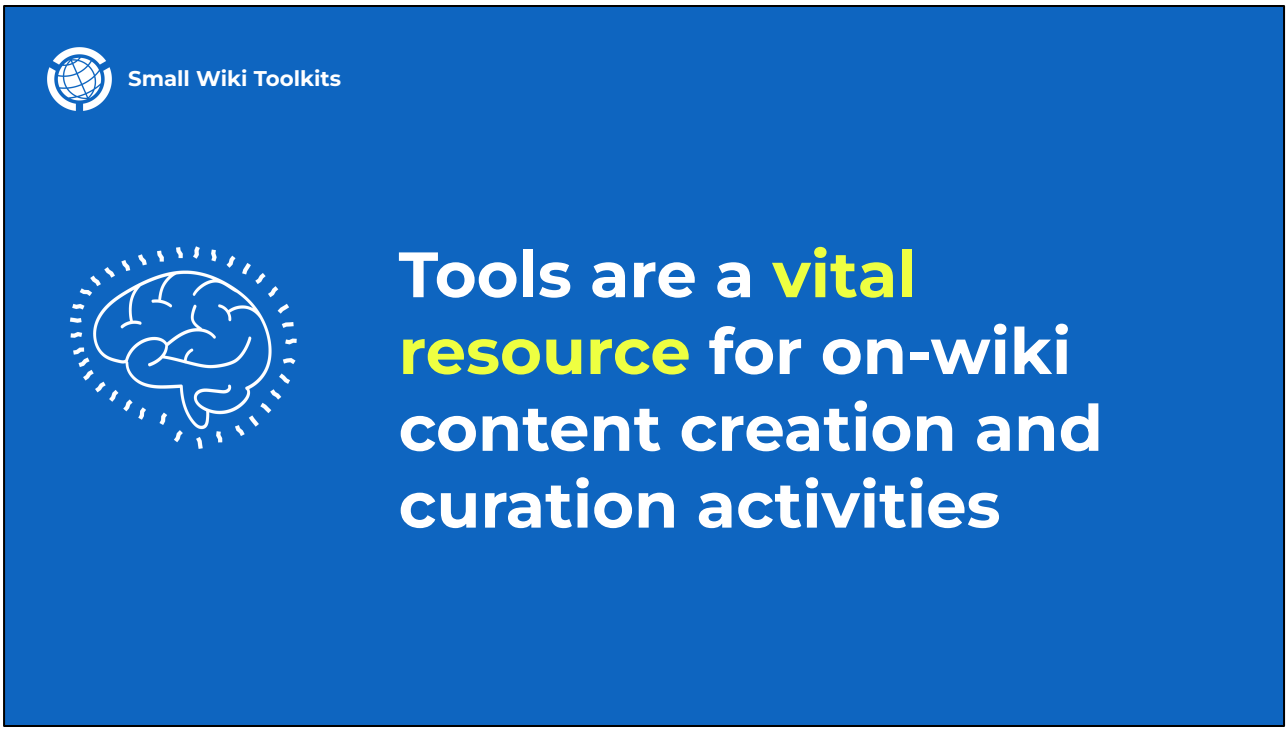
Small Wiki Toolkits

Best practices for maintaining bots and tools

[[User:BDavis (WMF)]]
August 2022

Hello everyone. My name is Bryan Davis and my pronouns are he/him. I am a Principal Software Engineer working for the Wikimedia Foundation as a member of the Technical Engagement team. I would like to talk with you today about some best practices that you can use when building and maintaining bots and tools for Wikimedia projects.



<read slide>

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 might say [citation needed]...

61.6%

Percentage of total edits made to Wikidata originating from tools and services hosts in Wikimedia Cloud Services, July 2022

Small Wiki Toolkits

More than 61% of the 24 million edits made to Wikidata during July 2022 were made by bots and tools hosted within Wikimedia Cloud Services (including Toolforge). This number was taken from the WMCS Edits Dashboard.

Small Wiki Toolkits

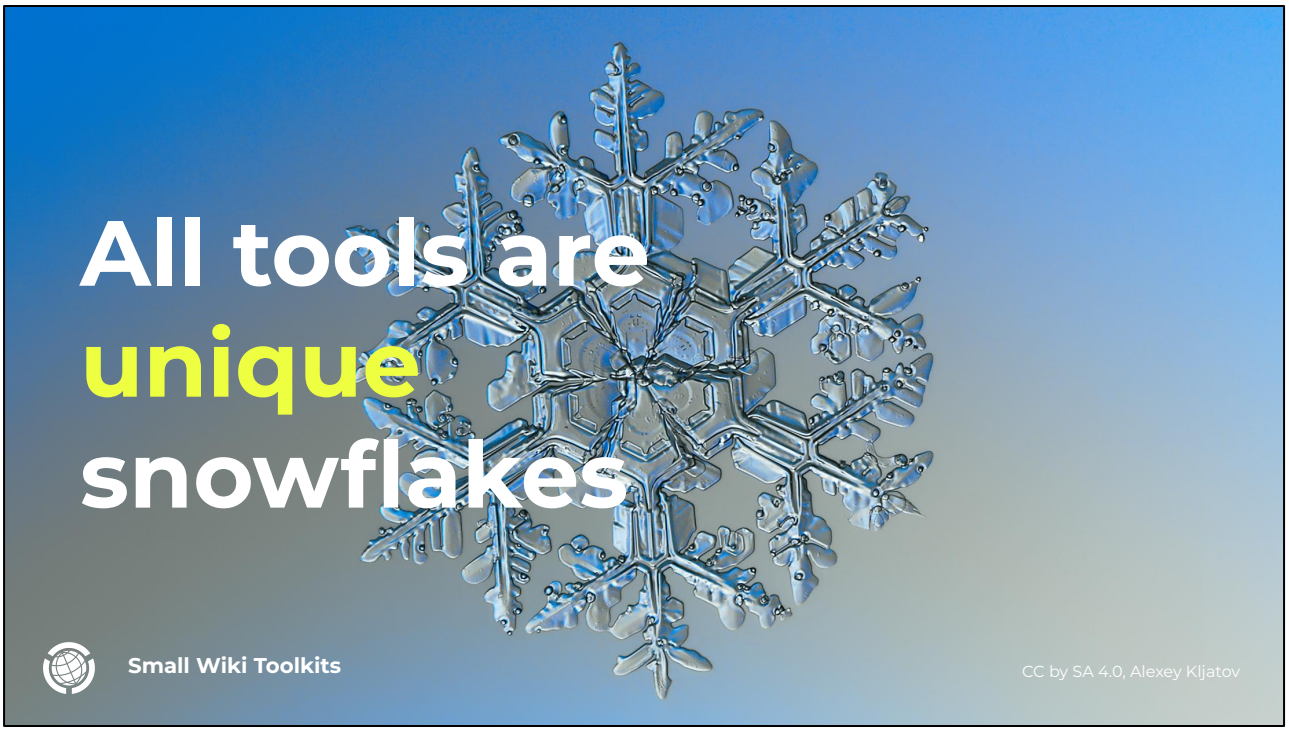
Time-to-revert by ClueBot NG's status

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 enwiki when ClueBot NG wasn't running.

One study from data that's 11 years old and a statistic about recent Wikidata activity doesn't prove my thesis, but hey at least I have some credible sources to point to. :)



A typical bot or tool project begins life as a way for a motivated Wikimedia community member to make some on-wiki task easier. 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 Wikimedia technical 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 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 Lévy and Sons

Small Wiki Toolkits

I have two real world examples of tools that had 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. The admins take on the work to migrate the tool to a newer runtime version and give it more memory to try and make it more stable. A community member takes on monitoring as a pet project and updates the ticket regularly when the tool is down. Admins 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 has a memory leak 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. Admins 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 responds. They explain that they have lost interest in maintaining 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". We tried a couple of times to get them to change their mind about declaring a license, 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 shut down 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 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.

* 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 maintainer responded that they would need Java 1.8 in order to fix their software. The maintainer is pointed to a previously declined Phabricator task for upgrading to Java 1.8.

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

* December 2015: Another user opens a new task requesting Java 1.8. This is investigated by Toolforge admins and again found to be a problematic upgrade at this time. When the new upgrade request is closed as declined, another bug specifically to look for a alternate solution for the tool is opened. 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. Knowing the maintainer was going to be away for awhile and that the deadline for closing the POST loophole is near, I used my Toolforge admin powers to look into how the tool was put together. I found more bad news: there is no source code on the Toolforge server, only compiled jar files. This greatly limits what anyone other than the maintainer 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 libraries used by the tool didn't work with the standard Java HTTP proxy configuration values. Further investigation found 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 and again asked the newer version of Java that had been rejected twice previously. They state that the bot is licensed under the Mozilla Public License, 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 failing out. One absentee maintainer, no source code, no license, procrastination, and demands for special treatment had killed the project.

Small Wiki Toolkits

FLOSS best practices for bots & tools

A tool that is valuable for an on-wiki workflow should protect the community by following best practices. As developers we put in a lot of effort to build new things and keep them running. No one should feel that they must be available 24/7/365 to support their tool. Popular tools will experience issues at all times of the day and night. By adopting a few simple practices common in the Free/Libre and Open Source Software ecosystem, a tool maintainer can make it easier for others to help them keep their tool running.

Best practices

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

Small Wiki Toolkits

These 5 core practices are what I would like all of you to try to follow with each and every tool you expect others to use. <read list from slide>

Let's talk about each of them in a little more detail...

Best practices

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

Small Wiki Toolkits

Pick a license

As many of you will know from your work in Wikimedia projects, copyright laws vary from country to country. In the United States and elsewhere, copyright automatically attaches to creative original works including software.

Without a license you are implicitly claiming copyright with no explanation of the rights you are willing to grant to others who wish to use or modify your software. Nobody else may reproduce, distribute, or create derivative works until standard copyright lapses without your explicit permission.

In the US today, copyright extends until 70 years after the creator's death. (That ridiculous duration will continue to grow too as long as Disney wants to protect Mickey Mouse.) If you think about how computers have changed in the last 70 years you will probably understand how this "protection" will make your software obsolete long before the copyright expires.

The Toolforge and Cloud Services Terms of Use say that you must use a license that has been approved by the Open Source Initiative. OSI is a US-based 501(c)3 non-profit responsible for maintaining the Open Source Definition standard. The OSD lists 10 criteria that all conforming licenses must comply with.

There are a number of different OSI approved licenses, and which to choose is largely a personal decision for the individual or team first developing a new tool. The two

easiest options for your license are:

- GNU General Public License v3.0 or later. This license ensures that all derivative works are made available under the same terms. It is conceptually similar to the Creative Commons Attribution-ShareAlike license that is used by most Wikimedia content projects. MediaWiki itself is licensed under an older version of the GPL.
- The MIT license. This license only requires derivatives to mention the original work and its license. This is conceptually similar to the Creative Commons Attribution license.

I can recommend https://choosealicense.com/ as resource for learning more about the differences between various licenses. Be aware the some of the licenses described there are not OSI approved however, so make sure to check against the OSI list before using a license for your project.

Best practices

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

Small Wiki Toolkits

Publish the code

Making the source code of your application public can feel scary, but without source code it is very difficult for the community to help rescue abandoned tools. It is ok to feel anxious about showing your work, but try to remember that most folks in the Wikimedia technical community have had the same feelings at some point. By sharing our code in public we can help others learn and maybe even get some feedback that will help improve our own code.

There are several gratis source code hosting options that tool developers can use on the internet. 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. Today (2022-09-18) those repositories will be created on Phabricator's Diffusion service, but I am working on a task to change the location to the newer https://gitlab.wikimedia.org/ service.

Best practices

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

Small Wiki Toolkits

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 maintainer to transition out of a project when they eventually lose interest.

A question that often comes up is how to actually find these co-maintainers. We will talk a bit more later about what I mean by "participate in the community", but for me this is the answer. Ask on mailing lists or the technical discussion spaces related to the wikis that your tool is designed to work with. Consider forming a mutual aid group with two or three other folks where you each become co-maintainers for everyone else's tools. Talk to your power users and the folks who most often notice when your tools are broken to see if they would be interested in learning how to help keep them working.

Best practices

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

Small Wiki Toolkits

Write some documentation

In 2004 Brion Vibber wrote as Bug #1 "Our docs are teh suck. Fix them up.". This is a problem for all software projects, including your tools.

A tool usually doesn't need a lot of documentation. Having a wiki page that explains how to start and stop the tool and a bit about how to troubleshoot common problems goes a very long way. We have the Tool namespace on Wikitech specifically for this kind of documentation, but put the docs wherever makes the most sense for you.

When writing these docs, think about the 5 most common things that can go wrong with your tool. What steps are needed to fix these issues? Go back and add to the docs when you discover new problems.

Some example documentation:

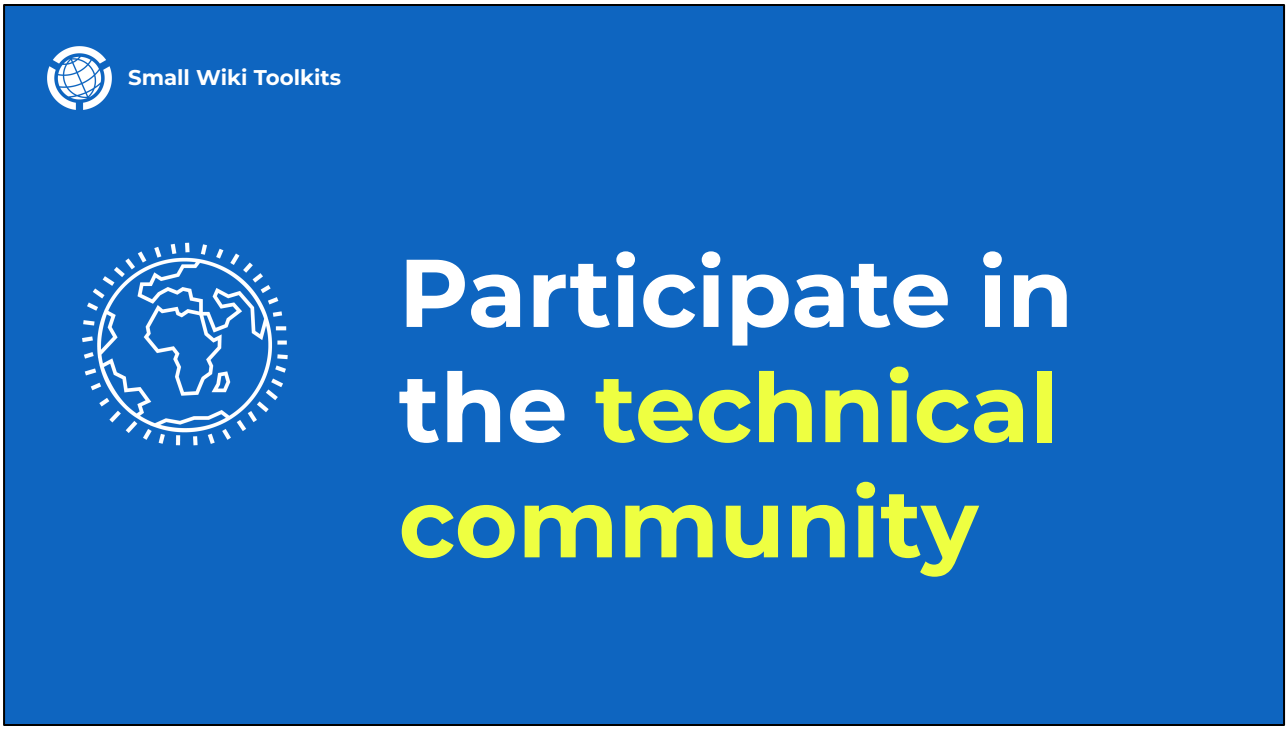
- https://wikitech.wikimedia.org/wiki/Tool:Bridgebot
- https://wikitech.wikimedia.org/wiki/Tool:Stashbot
- https://www.mediawiki.org/wiki/Wikibugs
- https://wikitech.wikimedia.org/wiki/Wm-bot

Best practices

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

Small Wiki Toolkits

Being here to participate in this workshop is a sign that you are already participating in the Wikimedia community, but let's talk a bit more about other ways to participate specifically in the technical community.



Participate in the technical community

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 scary, 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.

Technical community

➔ Join mailing lists:
- cloud@lists.wikimedia.org
- wikitech-l@lists.wikimedia.org

See https://lists.wikimedia.org/ for many more!

Small Wiki Toolkits

Mailing lists are one of the most "old school" community gathering places we have.

There are many mailing lists to discussing technical topics and receiving announcements about changes to software and services. Some of the lists I read and use most often include:

- *Cloud* which is for discussion of Cloud Services projects like Cloud VPS and Toolforge
- *Wikitech-l* which is for discussion of pretty much anything technical related to the Wikimedia movement
    - Speaking of old school, the archives for wikitech-l go all the way back to the first message to the list sent on February 8th, 2002 by Jimmy Wales.

There are many more specialized lists too. See lists.wikimedia.org for lists related to MediaWiki, the Action API, Dumps, Pywikibot, and lots more.

Technical community

- Libra.chat IRC channels
  - #wikimedia-cloud
  - #wikimedia-tech
- Telegram channels
  - Wikimedia Cloud Services support
  - Small Wiki Toolkits

Small Wiki Toolkits

In addition to mailing lists, there are a number of chat platforms and channels that are used by the movement and it's technical community.

IRC is another old school communications system still in use by the Wikimedia movement. There are a large number of Wikimedia channels on the libra.chat IRC network. The #wikimedia-cloud channel is a good place to ask for help with things specific to using Toolforge, Cloud VPS, and other Cloud Services products. The #wikimedia-tech channel hosts discussions on technical topics related to the Wikimedia wikis. There are many, many other channels for talking about specific services or wikis. [[meta:IRC]] is a good place to start looking for other channels and advice on using IRC in general.

We also use Telegram channels for some technical discussions. Some channels like Wikimedia Cloud Services support are bridged to IRC so you can use one or the other, but see all the conversations from either side. Others like the Small Wiki Toolkits channel are only on Telegram. [[meta:Telegram]] lists a few other technical channels as well as a large number of content project and thematic group related channels.

Technical community

- Use Wikimedia Cloud Services
  - Toolforge
  - Cloud VPS
  - PAWS
  - Quarry

Small Wiki Toolkits

The Wikimedia Cloud Services project is a collection of services designed to help the Wikimedia technical community test and host their tools.

Toolforge is a platform as a service environment providing access to a Kubernetes cluster and a legacy Grid Engine distributed computing environment. This environment has features specifically designed to make it easier for multiple maintainers to collaborate on operating a tool.

Cloud VPS is an infrastructure as a service environment similar to rackspace.com or AWS. When tools outgrow the Toolforge environment Cloud VPS can often be used to create a dedicated environment with more resources. The main trade off for the tool maintainers is that virtual machines created in Cloud VPS have to be maintained as well.

PAWS is a Jupyter notebook deployment hosted by Wikimedia. Anyone with a Wikimedia user account--the same account you use to edit the wikis--can use PAWS to run Jupyter notebooks. PAWS also has special features for using pywikibot to edit the wikis.

Quarry is another service that anyone with a Wikimedia user account can use. Quarry provides a web interface for running SQL queries against the Wiki Replicas, a set of live replica SQL databases of public Wikimedia wikis.

Small Wiki Toolkits

Thank you!

[[User:BDavis (WMF)]]
August 2022

Thank you for making the time to listen to me speak today. Does anyone have questions related to the talk that I can try to answer?

Credits

- Wikimedia Community Logo.svg By Artur Jan Fijałkowski, Public Domain
- Brain, By Jasmina El Bouamraoui and Karabo Poppy Moletsane, CC0
- 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
- MediaWiki, By Jasmina El Bouamraoui and Karabo Poppy Moletsane, CC0
- Wikisource, By Jasmina El Bouamraoui and Karabo Poppy Moletsane, CC0

Copyright © 2022, Bryan Davis and the Wikimedia Foundation.
This work is licensed under a Creative Commons Attribution-Share Alike 4.0 International license.