One year with GitLab
Catalyst’s Samba Team
Our journey today

GitHub

GitLab

Bootstrap
Background and statistics
Has it been it too hard to contribute to Samba?

Where did our new contributors go?
What happened to the students?
Many of us started on Samba as students
OpenHub (Ohola) contributor statistics:
By 2015 it looked pretty grim

https://www.openhub.net/p/samba/contributors/summary
Has Samba’s development slowed down?

Commits per Month

https://www.openhub.net/p/samba/commits/summary
Samba development is hard but rewarding

Samba is much more complex than when many of us started.
But with the right tools we can help new contributors start.
Being able to run a full test gives new contributors confidence!
My passion: That Samba be as welcoming and engaging as when I started
Accepting Samba contributions is hard

Review work is tedious
Particularly if done well!
Also much less enjoyable if rework required:
To many e-mailed patches didn’t actually compile or pass CI
Mechanics still quite manual
Even with GitLab (so far)!
sn-devel / autobuild limits

- Samba-team only
- Single distribution (a single Ubuntu version per release)
- Hardware limitations
- Many simultaneous runs cause load spikes and failures
- Catalyst engineers had been using the Catalyst Cloud since 2015
- ‘start-samba’ Script to run autobuild on a ephemeral VM
- But this is still not a general solution
GitHub

Or the long winding road so far
Samba: Solving social problems with technical solutions since.... (forever?)
2015: It all started with GitHub

Worried about a dip in contributions
GitHub pitched as an alternate contribution mechanism
Concerned that the mailing list focus was putting off potential contributors
GitHub seen as ‘Where all the cool kids are’
Samba Team official GitHub mirror established
2016: Travis CI

Dipping toes into CI for contributors

Test results for some of make test shown with the pull request

Good at finding simple issues but not a full test
2017: Pushback and lessons learned

GitHub never embraced by the core Samba Team
Even I didn’t use it for my own code
Not a free software solution
One-way GitHub -> mailing list script very annoying
Until it broke
I was the owner but did not follow up on that
Pull requests left ignored for years
It really matters that the team use the same tools as new contributors

New developers should be able to trust the contribution instructions!
GitLab

Learning the lessons of GitHub
2017: Preparing for GitLab

Thanks to Catalyst and clients, resources found to experiment with GitLab

Joe Guo automated a cloud-based ‘runner’

Autoscaling by launching a new server for every task

Jamie McClymont split up our selftest into parallel parts

Therefore faster runtime (in parallel)
2018: GitLab CI introduced

A simple way to pre-test commits
(Before pushing to autobuild)
A full ‘make test’ after each ‘git push’
Low key introduction
Patches still on the mailing list
Just with a CI link included
Key feature: Non team access

Samba contributors given access ‘by request’
Remove barriers between ‘team members’ and ‘new contributors’
SambaXP 2017 was key
Signed up most of the Samba team around the e-mail room
Merge requests

Reviewers can see the patch, discussion and CI results at the same time
Samba’s wiki still said ‘send a GitHub pull request’
It was requested that we switch all references to GitLab
So merge requests became the documented behaviour!
But really, it happened organically
Many team members were already submitting merge requests
Lessons learned

Team engagement is vital!
Genuinely practical ‘hook’ of CI while sleeping
Compared to staying up late watching sn-devel to say ‘it passed’
Allowing others to suggest the logical next step shares ownership
Software-as-a-service and cloud helped a lot
This avoided tricky Samba Team investment discussions up-front
Autobuild / GitLab CI in 1:50: Thanks metze!
Bootstrap

Testing more than Ubuntu 14.04

Started by Joe Guo
Bootstrap: A Samba dependency management script

Reliably answering the question:

“What packages are required to build Samba?”

Authoritative for a source build

Stored in git

So it is the list for this version, not a packaged version

Linked from the wiki

Replaces the hand-maintained lists (eventually)
Keeps distribution package lists consistent

Strongly discourages adding a dependency for just (say) Ubuntu

Helps ensure we enable the same features on all supported platforms:

- Debian 9
- Ubuntu 16.04
- Ubuntu 18.04
- OpenSUSE 15.0
- CentOS 7
- Fedora 29
Generates container images

- Allows Samba to be tested against multiple distributions
- Uploads into GitLab's Docker registry
- Proves we can build on (eg) CentOS 7 with Python 3.6
Infrastructure as code

- Bootstrap is entirely in samba.git
- Full commit history on changes to the containers used for CI
- Allows restart on another GitLab
- Pull the images between registries
- Regenerate from the git commits
- Sync with sn-devel is still manual however (speak with root@)
Lessons learned

We can build on GitLab

We are willing to go beyond ‘simulated sn-devel’

Team engagement is unpredictable but incredibly worthwhile

Initially written off as ‘too complex’

Turned out to be barely complex enough

A really elegant solution focussed around git

Robust locking between image list and CI image used!

SHA1 of all relevant files put into image name
Ownership by bikeshedding!
In conclusion
GitLab / Bootstrap: a success?

Python3 migration successfully achieved
Most py3 patches went through GitHub and then GitLab
Bootstrap gives confidence we can ship Python3-only on CentOS 7
It was a success for my efficiency:
Reviewed 1700 patches in the past year!
Previously I did about 1000 per year
Seems to be slightly more contributors this year
Another type of success: Change without a team crisis!
How do we replicate it?

Creating a culture of experimentation is hard

Samba is a consensus-based community

How to try things that may not work?

How to build up a critical mass of users in an opt-in culture?

If the experiment is about external engagement it is hard to run without the team on-board first

How to find the resources to invest in the experiments?

Hopefully the next changes can be more evolutionary than revolutionary
Managing Change is hard everywhere

But Samba has a very strong resistance to changes in work patterns.

Team ‘lore’ is that the only time to change a VCS is when Jeremy is on a intercontinental flight!

I strongly resisted the git change for Samba4!

How to find the line between:

Encouraging change

The style of coercion that leads to resistance rather than cooperation.
Change is a cost to those being changed
What next?

I would prefer not to drive the next change
So Change in Samba doesn’t just become an ‘Andrew’ or ‘Catalyst’ thing
Hopefully a more ‘agile’ team can make the next changes with less resources
Better coordination with the root@ team
These changes specifically avoided asking for anything other then Rackspace access
Ideas I’ve heard

Automate release note creation
Make applying backports less work intensive for Karolin
These can be uploaded to bugzilla but not apply or pass
Testing beyond Samba: run cifs kernel client etc
Tests on FreeBSD
2 Factor Authentication

GitLab supports U2F tokens

Require for Samba Team?

https://commons.wikimedia.org/wiki/File:FIDO_U2F_Security_Key_by_Plug-up_International_02.jpg
My suggestions regardless

Have sn-devel trust GitLab CI instead of local autobuild execution?

Some day someone will be embarrassed when a task passes on sn-devel that fails GitLab CI

Consider “marge-bot” or similar to do merges on GitLab?

Can we make back-ports and security process less work-intensive to focus:

more on thinking, less on clicking

Could we use merge requests for backports?

Bugzilla 6.0:

Multiple attachment uploads
Any Questions?

abartlet@catalyst.net.nz
abartlet@samba.org

https://catalyst.net.nz/samba-%26-windows-integration