




Growing GitLab DevSecOps platform to +40k users

Krzysztof Walkiewicz

Who am I?

Krzysztof Walkiewicz

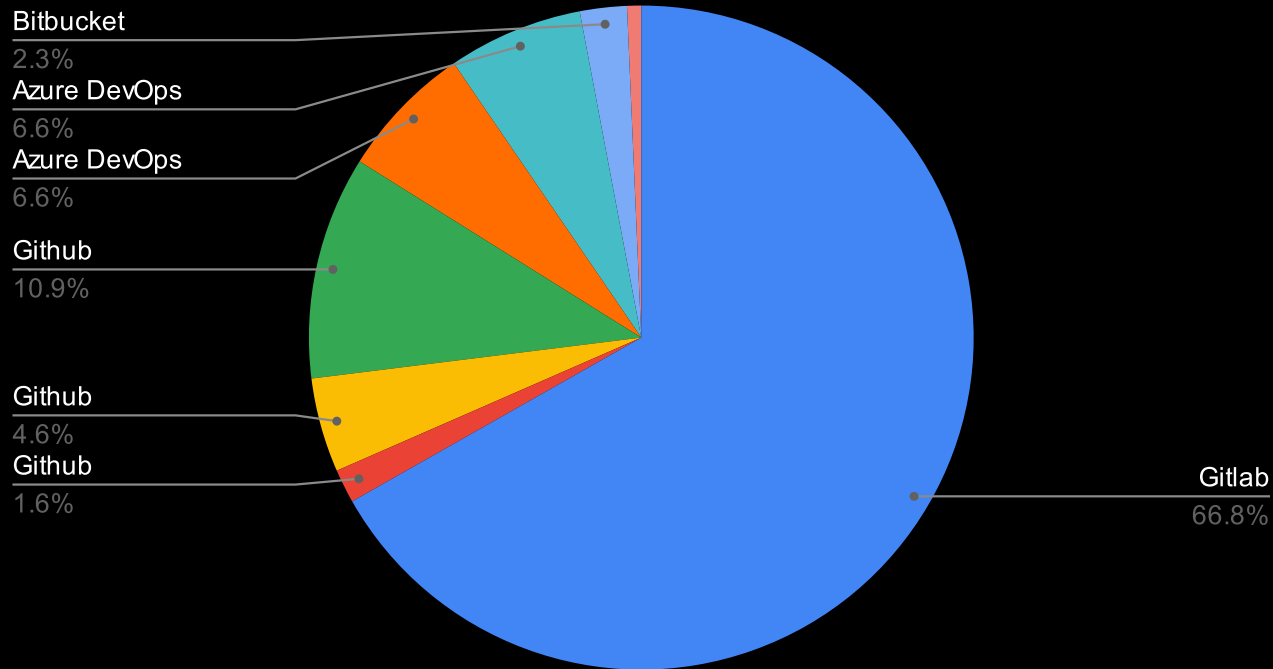
-  DevOps engineer
-  Product owner of DevSecOps platforms @ Roche
-  Today I'll be your storyteller

What is this talk about?

- This is a story about:
 - a great team
 - open/inner source
 - passion for the technology and the fun factor
 - empowerment and trust
 - confronting status quo
 - DevOps culture

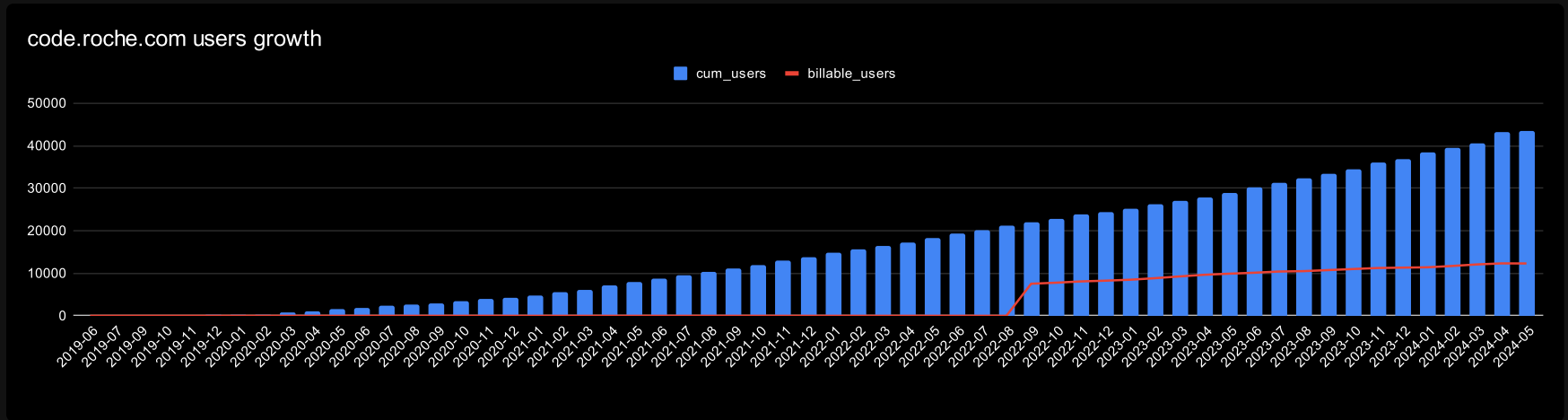
Current state

DevSecOps platforms market share @ Roche - 2024-04



User base growth

What counts is a developer head count 😊



Time travel to 2019 and the need for change

How did the developer experience look like @ Roche back then in the context of tools used in software development?

We had - and still have many tools:

- Bitbucket, Github, SVN, ADO, Jenkins, Bamboo, Jira...
- All of these as **multiple instances** (on-prem and the cloud) across the organisation.
- "*Guarded*" behind access requests (which might have been easily rejected if you didn't have good enough business case).
- With different (sometimes lacking) capabilities, often limited by the license.

The need for change (continued)

- Many support teams maintained different part of the stack - but none of them had holistic view.
- Many silos around them.
- There was no coherent strategy for inner source/reusability inside the organisation, with no insight into global codebase.
- As a result different teams were *re-inventing the wheel* over and over again.

Holding out for a hero 🤔

On March 14, 2019 one individual decided to change the reality and made an *initial commit*. Slowly, over time, strong team of enthusiasts formed around the initiative of having modern DevSecOps platform that would support DevOps and Inner Source.

- The inspiration came actually from our hosts - Siemens 🧑
- It was totally bottom-up initiative.
- Without typical approval-requirements-vendor-assessment-funding-phase (or in whatever order this usually happens).
- The project was actually never announced - it grown organically.
- Access to openstack deployment was a turning point ("*I can handle the network!*" - unbounded creativity).
- "Business done this because IT failed".



What were the success factors of introducing GitLab @ Roche?

Given that many of IT projects:

- Fails to deliver a value.
- Are not delivered on time.
- Exceeds budget.
- Bootstrapping new platform inside a big organization it's like running a startup - (hint: it's rather risky).

So what happened at Roche? Let me answer this question.

Success factors

Great team

- **Small** - size matters.
- **Skilled**, ambitious, motivated.
- **Empowered** (ownership of the whole solution: provision, maintainance, configuration, monitoring, support).
- Following good engineering practices:
 - Code reviews.
 - Infrastructure as code.
 - Automated tests, validation as code.
 - A lot of documentation.
 - Postmortems.

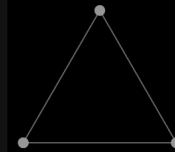
Language	Files	Lines	Blanks	Comments	Code	Complexity
Terraform	254	13582	1821	1001	10760	801
Markdown	230	19195	4922	0	14273	0
YAML	108	21583	732	308	20543	0
Python	51	4175	743	196	3236	215
Smarty Template	44	2868	222	0	2646	99
JSON	43	104864	34	0	104830	0
Shell	32	1742	322	243	1177	125
Go	21	1574	198	94	1282	258
Plain Text	11	44	0	0	44	0
Gherkin Specificati...	7	631	50	2	579	31
gitignore	5	43	8	8	27	0
JavaScript	3	57	7	10	40	8
Jinja	3	136	20	0	116	3
TOML	3	51	9	0	42	0
CSS	2	423	12	4	407	0
INI	2	15	1	0	14	0
XML	2	52	0	0	52	0
BASH	1	5	1	1	3	2
C#	1	12	1	0	11	0
Go Template	1	48	12	0	36	5
HTML	1	78	0	1	77	0
Java	1	13	1	4	8	0
License	1	201	32	0	169	0
MSBuild	1	14	4	0	10	0
PHP	1	11	2	0	9	0
SVG	1	4	0	1	3	0
Total	830	171421	9154	1873	160394	1547

Estimated Cost to Develop (organic) \$5,585,252
Estimated Schedule Effort (organic) 26.44 months
Estimated People Required (organic) 18.77

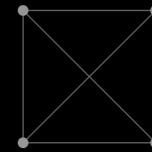
Success factors (continues)

Good engineers to facilitators ratio 😊

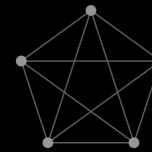
Role	Primary tool
DevOps engineer	Vim
DevOps engineer	Vim
DevOps engineer	VS code
DevOps engineer	VS code
DevOps engineer	Emacs
DevOps engineer/PO	Emacs
Validation lead	Gdoc
Manager	Gmail



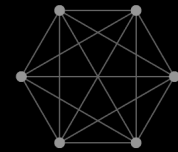
3 people, 3 lines



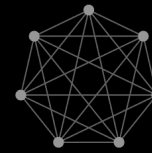
4 people, 6 lines



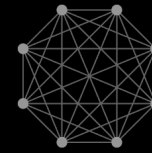
5 people, 10 lines



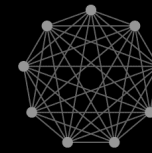
6 people, 15 lines



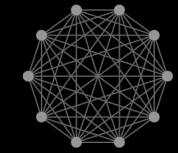
7 people, 21 lines



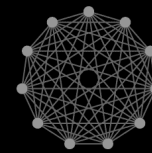
8 people, 28 lines



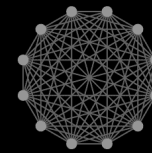
9 people, 36 lines



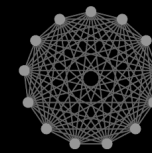
10 people, 45 lines



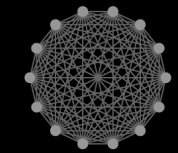
11 people, 55 lines



12 people, 66 lines



13 people, 78 lines



14 people, 91 lines

Success factors (continues)

GitLab is simply a great product ❤️

- With complete set of features (even on the free version).
- With server/cloud installation options.
- Helm charts available.
- Great documentation and a handbook!
- Developed in the open (saved us many times).

Success factors (continues)

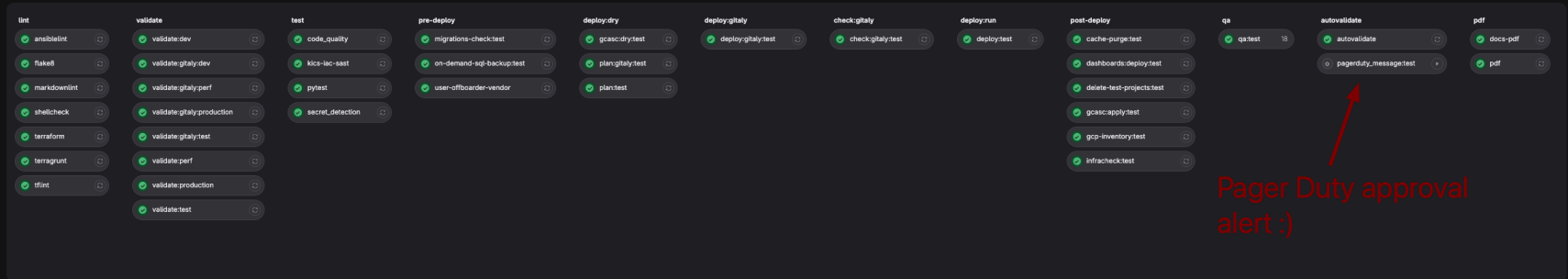
Google Cloud - delivers quality products and an API for them, enables hyper-automation

- GKE
- CloudSQL
- Cloud storage
- Monitoring

All these great products helped us build upon and quickly deliver value to the end users.

Success factors (continues)

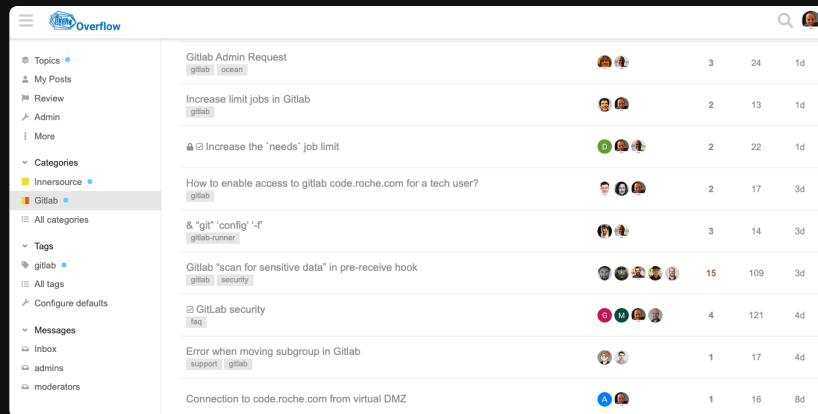
Dog-fooding: platform deployed from within GitLab CI/CD pipeline (yes, inception)



Success factors (continues)

Support to the users delivered via community driven forum

- Acts as a knowledge base.
- Users can search for similar problems.
- More experienced platform users (from outside of core team) started sharing knowledge.
- We may refer to the old answers (DRY).



Success factors (continues)

Courageous and visionary leaders

- *"There were people with the vision, that saw a potential and took the pressure".*
- *With "It is easier to ask for forgiveness than to ask for permission" approach.*
- *Not afraid of confronting Status quo.*

Lessons learned (people aspects)

- It's good to be conservative when it comes to specifying platform limits. Example: 50MB max push size is ok. Saying definitive "no" is also ok - if you can justify it.
- Be transparent.
- Gather like-minded people around if you try to challenge status quo in the organisation.
- Maintain fun factor within the team.
- People come and go - that's ok - plan for this.
- Iterate fast, gather users feedback - it's really important.

Lessons learned (technical aspects)

- There are no such things as infinite resources (avoid `no limits` settings).
- Complex platforms are highly dynamic - *Panta Rhei*.
- Pin all the dependencies - there is enough randomness in the Universe.
- Every codebase is aging quickly and need mainenance. It's not an asset, it's liability.
- Laverage scalability features of Kubernetes.
- Accurate monitoring is essential (focus on golden signals: latency, traffic, errors, saturation).
- Have an eye at Queue wait time of CI/CD jobs in the runners fleet.
- Read deprecations and release notes carefully.
- If you deploy to Kubernetes - make sure you have at least 2 CKA's in the team 🤔
- Uptime of 99.9% is doable. Everything above is rather difficult and expensive.

Thank you!

1. Images on page 1 and 8: unsplash.com

2. Team complexity growth image on page 11: <https://getlighthouse.com/blog/developing-leaders-team-grows-big/>