GitLab

7 DevOps steps you're missing — and how a DevOps platform can help

It’s time for DevOps without excuses. Teams know what they should be doing but because of time constraints, tech hurdles and information silos the “extra” steps often don’t happen. Here’s how a DevOps platform can make everything easier.

Table of contents

Introduction

It's a truth universally acknowledged: DevOps teams know what they should be doing, but they often don't do it. In the rush to get software out the door, "nice to haves" – like extra testing, monitoring, or tighter feedback loops – are passed over in favor of "get it done." Like diet and exercise, DevOps isn't easy.

That doesn't mean you can't push your DevOps practice to do more. We're going to look at seven potentially overlooked steps and show you how a DevOps platform would make them possible.

A unified DevOps platform eliminates the time lost to piecemeal integrations allowing teams to extend their reach into extra processes that will have almost immediate payback. **"By 2023, 40% of organizations will have switched from multiple point solutions to DevOps value stream delivery platforms to streamline application delivery, versus less than 10% in 2020," predicted Gartner** in a Strategic Planning Assuption found in its Market Guide for DevOps Value Stream Delivery Platforms*.

Gartner, Market Guide for DevOps Value Stream Delivery Platforms, Manjunath Bhat, Hassan Ennaciri, Chris Saunderson, Daniel Betts, Thomas Murphy, Joachim Herschmann, 28 September 2020

Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

With a DevOps platform in place, teams will have the technology, tools, and even time to implement best practices including:

- » **Better visibility and traceability** that improves troubleshooting and tightens feedback loops.
- » **Dramatically more testing**, earlier in the process, can speed up release times and boost code quality.
- » **Performance monitoring** lets everyone sleep better at night.
- » **Increased collaboration** with the business side means getting functionality right the first time.
- » **Simplified compliance tracking** eliminates countless headaches.
- » **Reduced technical debt** isn't just a goal, it can be a reality.
- » **Controlled deployments** are a potential game-changer for some teams.

Making the case for more

In our 2020 Global DevSecOps Survey we heard from developers about tasks they're not doing now, thanks to DevOps, including manual testing, manual deployments, code syncing, ticket creation, and more. But they also had a long list of things they know they should be doing, but aren't: More testing of all types, from A/B to unit and security, increased monitoring, more and better code reviews, and increased use of cutting edge technologies including containers.

Why aren't they doing more of what they know should be doing? The short answer is: It's too hard and/or too time-consuming. That's where a DevOps platform comes in to streamline all the things. If everyone involved in the software development lifecycle used a single DevOps platform, more will be accomplished because there is just less of everything that gets in the way – less tool integration, handoffs, communication platforms, information silos, and required manual interventions.

1. Bring traceability to incident management

Incident management is an unavoidable (if adrenaline-producing) part of every DevOps team, so it's critical to find a way to streamline the process. Typically, monitoring tools exist separately from the IDE and the scanning and deployment processes, so alerts are external, not linked with the go-to tools, and there is no single place to find all the relevant information.

With a DevOps platform in place, teams can easily pull data from incident management tools on board, receive notifications of alerts, triage alerts and incidents, and keep key stakeholders in the loop all without ever leaving the platform. A single platform means lightning-fast feedback loops and no one will have to go hunting to find key information.

It's important to remember that a DevOps platform makes incident management bi-directional; it's just as straightforward for developers, security, and ops to see a problem as it is for post-production incidents to be flagged, alerted, and brought into the same system. In fact, the ability to go backwards when diagnosing a problem is invaluable, particularly if everyone who needs access to the key information can easily get it.

Since the time frame from identifying an issue to diagnosis and remediation is very tight, a DevOps platform that's in lock step with an incident management solution means faster resolution because teams have visibility from the meta level and don't have to go looking elsewhere for information. By itself visibility, isn't sufficient. However, speedy incident management also relies on communication and collaboration, both of which are made dramatically easier by the use of a single DevOps platform.

If there's a case to be made for your team doing more with incident management, it's this: Busy DevOps practitioners put out the fires, but do they have the time and tools to do more than that? Tracking "near misses" is a great place to start – the Aviation Safety Reporting System has been doing that since 1976 and flight-based fatalities have dropped by 65% since then. A unified DevOps platform surfaces the data so even busy teams have time to analyze **all** of the data, including near misses and the ratio of problems between pre and post production.

2. Test everything

Does any DevOps team do enough testing? Based on what our survey respondents said, the answer is a hard no. A full 47% of those surveyed pointed to test as the number one reason for delayed code, and when asked about what their teams should be doing more of, the vast majority said testing of all kinds.

> **"Too little testing done too late."**
>
> **"We are slow and do not test very well. We do Big Bang deployments."**
>
> **"No designated tester so the developer or whoever gets time deals with it."**
>
> **"Not enough tests (or none) and then the code doesn't work in production. Some collaborators have poor IT skills."**

Additional survey data shows most teams aren't regularly running static application security (SAST), dynamic application security (DAST), compliance or dependency scans. And even those teams running those scans acknowledge that most of the time the results don't come back to them in their IDEs.

To resolve the testing quandary, start with a comprehensive DevOps platform. A single space where everyone works will make it easier to shift testing left (aka earlier in the process) and developers won't have to go hunting for results as they'll be delivered right in the merge request. If automated testing is built into the DevOps platform, it will be easier; and the easier it is, the more testing that can happen.

Also, one of the biggest challenges in testing are edge cases or exceptions, and a DevOps platform can help with that too. By automating the planned tests, developers and testers have freed up time (and energy) to create custom/manual tests for anything that's out of the box.

Everyone knows testing is only half the battle – once bugs are found they have to be fixed. Again, a unified DevOps platform offers built-in visibility and alerts that will streamline this often very annoying step in the process. The earlier bugs are found in the process, the more likely developers will fix them, meaning security and ops aren't having to deal with this in a kludgy way much later in the process. Also discovering a bug right after a commit means other devs are less likely to build on faulty code.

3. Mainstream performance monitoring

Like testing, performance monitoring is another step in the development process that doesn't happen often enough because the tools to do so aren't baked into the process. There's no arguing that understanding how code actually performs "in the wild" is useful for developers and takes the load off of operations, but who wants to stop the workflow, reach for another tool, and run simulations? The only other option would be to wait for QA and that's not a great solution either.

A DevOps platform solves that problem and gives busy developers the opportunity to get real world insights into their code – easily – before it hits production. A unified DevOps platform lets a developer "browser test" new code using review apps. Browser testing, the related load testing and DAST are critical steps to take, particularly in environments where code is being shipped out the door quickly; there's tremendous satisfaction in being able to actually see that something is working as expected in the real world.

4. Collaboration that works

In our 2020 Survey we asked developers, ops pros, security team members, and testers what would be the most important skill for their future careers. Their answers were unanimous: Communication and collaboration will be the most vital going forward.

Everyone knows how important it is to communicate and collaborate, but making those things easy can be a challenge. Geographical distance is a huge hurdle, but an even bigger challenge is to ensure that everyone has access to the same information, no matter what role they're in. In many organizations, collaboration tools can be scattered and completely divorced from where the actual work gets done.

> **A unified DevOps platform brings all stakeholders to the same table whether it's product management, security, HR or legal. If everyone uses the same platform, it's easy to comment, share, and have true visibility into every part of the software development process.**

All team members can see the problem and the problem is also actionable – it can be fixed by commenting or improving, reassigning, or through suggestions at the point of discovery in just a click or two. Finally, integrate a chat app into the DevOps platform and suddenly the entire org is painlessly connected.

There's one more significant benefit of increased collaboration and communication: Do it right using a DevOps platform, and friction points and misunderstandings will begin to ease. Developers often think "the business side" simply doesn't get it, and it's safe to say the business folks think devs are too far removed from the balance sheet. Collaboration around a single platform will mean everyone learns new-to-them lingo that will improve communication and outcomes moving forward.

5. Compliance made easier

Keeping track of rules and regulations is mandatory in a number of industries and a best practice in most others. But compliance is, to put it frankly, a huge headache in most organizations largely because the tools used to track it are all over the map. Even large companies are still using spreadsheets, emails, and Google drives to oversee compliance efforts. Imagine having to go hunting for data in all of those different places and then having to tie it into a third-party tool or service. Take this one step further – actually into the DevOps team – and it becomes clear why compliance teams often can't keep up, let alone scale.
Also, many organizations don't have a single compliance silo; some have added Governance, Risk, and Compliance (GRC) tools separately, and those also don't integrate with DevOps. In fact, dev and ops tend to view compliance-related tasks as "getting in the way" of speedy releases and that's not helpful either. The bottom line: Compliance is so hard and so divorced from the software development lifecycle that it's probably the last thing a DevOps team wants to consider.

A unified DevOps platform can change that. If everyone is on a single platform it's easy to define user roles and permissions down to a granular level so credentials, security scanning, and approval rules can be established. Since application security is built into the platform, it's a simple matter to automate InfoSec compliance requirements. Certain industries have very specific compliance requirements (like HIPAA or SOX) and working on a single DevOps platform makes it easy to track collaboration, chain of custody, and overrides through issues and merge requests - there's no need to have a separate tool to manage this process. And, finally, even audits will be more straightforward since there's only one place to look for user actions, permission changes, logins, etc.

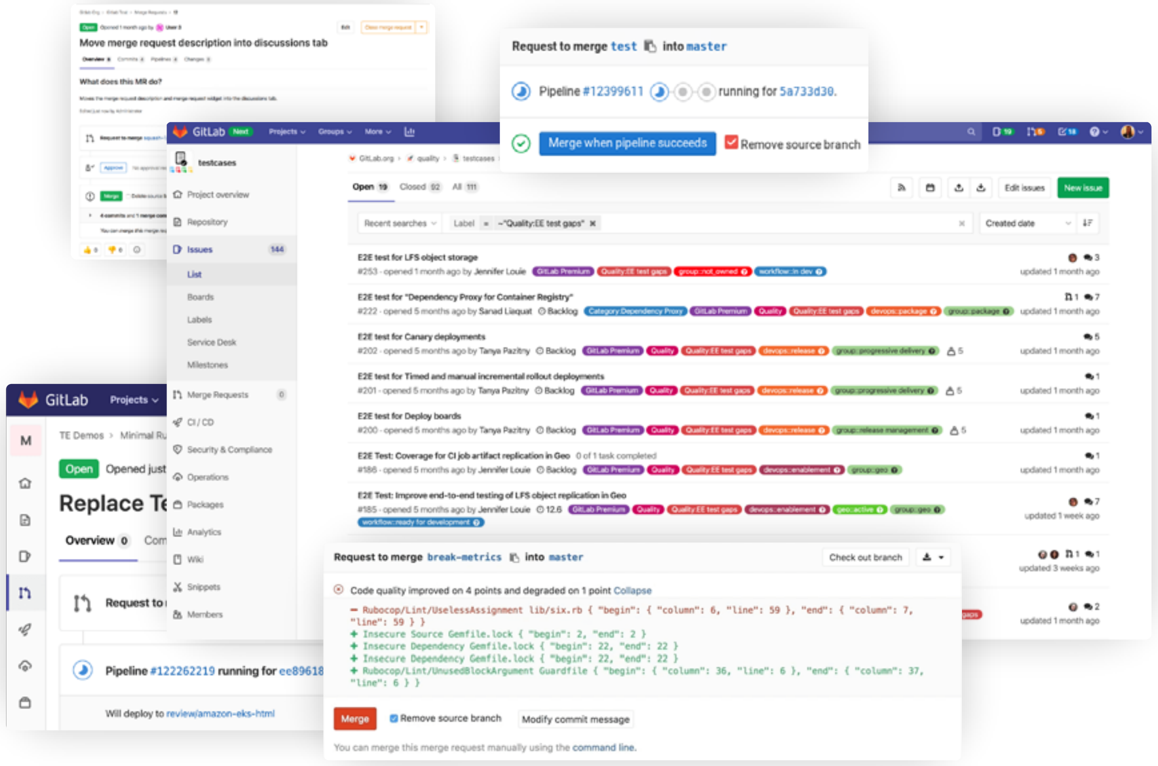
6. Turn technical debt on its head

Every DevOps team struggles with technical debt, beginning with justifying the time to address it. A DevOps platform can help that, though, because for the first time non-DevOps practitioners (i.e., product owners/the business side) have access to data they've probably never seen before.

When the business side is wondering why a DevOps team is spending time on technical debt, a look at the numbers could show that 25% of deployments are failing and the mean time to recovery is six hours...in other words, there's value in spending time paying off technical debt. A DevOps platform can become a way to pitch the process, give more context to business owners, and even give dev and ops a bigger seat at the table.

7. Take advantage of controlled deployments

Every DevOps team wants more automation with more control and the ability to deploy as much or as little code possible. But those same busy DevOps teams often have to settle for all or nothing. A DevOps platform can change that. By using just one tool, it's easy to bring all the disparate pieces of controlled deployment together in one place, from multiple approvals to authentication, authorization, and ensuring the right people are working on the right set of tasks. Suddenly dev and ops have more control over their DevOps processes than they've ever had before, particularly if they're moving from multiple tools to a single tool.


Request to merge test into master

Pipeline #12399611 running for 5a733d30.

Merge when pipeline succeeds ✓ Remove source branch

Open 19 Closed 92 All 111

Label = ~"Quality:EE test gaps"

E2E test for LFS object storage
#253 · opened 1 month ago by Jennifer Louie GitLab Premium Quality:EE test gaps group::not_owned workflow::in dev — updated 1 month ago

E2E test for "Dependency Proxy for Container Registry"
#222 · opened 5 months ago by Sanad Liaquat Backlog Category:Dependency Proxy GitLab Premium Quality Quality:EE test gaps devops::package group::package — updated 1 month ago

E2E test for Canary deployments
#202 · opened 5 months ago by Tanya Pazitny Backlog GitLab Premium Quality Quality:EE test gaps devops::release group::progressive delivery — updated 1 month ago

E2E test for Timed and manual incremental rollout deployments
#201 · opened 5 months ago by Tanya Pazitny Backlog GitLab Premium Quality Quality:EE test gaps devops::release group::progressive delivery — updated 1 month ago

E2E test for Deploy boards
#200 · opened 5 months ago by Tanya Pazitny Backlog GitLab Premium Quality Quality:EE test gaps devops::release group::release management — updated 1 month ago

E2E Test: Coverage for CI job artifact replication in Geo 0 of 1 task completed
#186 · opened 5 months ago by Jennifer Louie Backlog GitLab Premium Quality Quality:EE test gaps devops::enablement group::geo — updated 1 month ago

E2E Test: Improve end-to-end testing of LFS object replication in Geo
#185 · opened 5 months ago by Jennifer Louie 12.6 GitLab Premium Quality Quality:EE test gaps devops::enablement geo::active group::geo workflow::ready for development — updated 1 week ago

Request to merge break-metrics into master

Code quality improved on 4 points and degraded on 1 point Collapse

```
− Rubocop/Lint/UselessAssignment lib/six.rb { "begin": { "column": 6, "line": 59 }, "end": { "column": 7,
"line": 59 } }
+ Insecure Source Gemfile.lock { "begin": 2, "end": 2 }
+ Insecure Dependency Gemfile.lock { "begin": 22, "end": 22 }
+ Insecure Dependency Gemfile.lock { "begin": 22, "end": 22 }
+ Rubocop/Lint/UnusedBlockArgument Guardfile { "begin": { "column": 36, "line": 6 }, "end": { "column": 37,
"line": 6 } }
```

Merge ✓ Remove source branch Modify commit message

You can merge this merge request manually using the command line.

Pipeline #122262219 running for ee89618

Will deploy to review/amazon-eks-html

Try GitLab free

All GitLab features — free for 30 days.
GitLab is more than just branching strategies. It is a full software development lifecycle & DevOps tool in a single application.

Start your free trial

About GitLab

GitLab is a DevOps platform built from the ground up as a single application for all stages of the DevOps lifecycle enabling Product, Development, QA, Security, and Operations teams to work concurrently on the same project.

GitLab provides a single data store, one user interface, and one permission model across the DevOps lifecycle. This allows teams to significantly reduce cycle time through more efficient collaboration and enhanced focus.

Built on Open Source, GitLab leverages the community contributions of thousands of developers and millions of users to continuously deliver new DevOps innovations.

More than 100,000 organizations from startups to global enterprises, including Ticketmaster, Jaguar Land Rover, NASDAQ, Dish Network, and Comcast trust GitLab to deliver great software faster.

GitLab is one of the world's largest all-remote companies, with more than 1,250 team members in more than 65 countries and regions.

Learn more at GitLab.com

GitLab