6 ways to measure the ROI

of an Automated Software Delivery Solution

GitLab

SINCERA
Accelerating Digital Transformation

Introduction

It's possible for your organization to unlock more business opportunities and unleash your team's productivity; what it takes is Automated Software Delivery. Automating software delivery has gone from being a desirable addition to your Software Development Life Cycle (SDLC) to a prerequisite for success — not just for your customers' sake, but also your teams' productivity and job satisfaction. With speed of delivery becoming more and more important in an increasingly competitive landscape, it has become critical to move from manual to automated, from tactical to value creating work, from siloed teams to collaborative work environments.

GitLab

SINCERA
Accelerating Digital Transformation

What’s inside?

What does success look like?

We believe automating your software delivery allows your organization to:

1. **Automate or eliminate manual, repetitive tasks** which improves your developer's productivity and job satisfaction
2. **Boost the overall velocity of your software factory** allowing for faster releases, greater competitive advantage and increased market value
3. **Strengthen collaboration** across development and operations teams allowing you to improve efficiency of your SDLC
4. **Modernize your applications** to give you the flexibility to adapt faster to your customers' requirements

"We selected GitLab to automate and integrate processes during the deployment process. Previously, the majority of things were done manually and now the team wants to follow DevSecOps philosophy and governance. We wanted a DevOps platform with state-of-the-art CI/CD capabilities, the ability to quickly onboard new developers, and features to improve compliance and monitoring functionality. Now, Gitlab is one of the core components of our systems."

— Principal product manager at an **organization that maintains essential Internet infrastructure in France**

ROI of automated software

When you've already invested a lot of time and money into getting your existing investments to work, considering another solution for Automated Software Delivery could feel disruptive – unless you consider the return on investment (ROI). Many organizations struggle with what to measure and how to measure it.

ROI is often measured in savings and revenue, but this approach can have limitations on an organization's true ability to see their full ROI. There are tangible and intangible returns from automating software delivery which must be calculated.

In this guide, we'll walk you through 6 key considerations — both tangible and intangible — for calculating the ROI of adding an Automated Software Delivery solution.

Gains in productivity

Automated Software Delivery can help minimize repetitive tasks and improve team member productivity. Time previously spent in manual processes (e.g., setting up a test environment, running testing scripts, integrating with various other components of their software, waiting for review comments) can be leveraged for generating business value like new functionalities, product updates, and quality, performance, or efficiency improvements.

"One obvious way the developer experience has improved is that things are moving more quickly. Their previous CI/CD solution took between 35 to 45 minutes to get from commit to staging, but GitLab comes in reliably at just 13.5 minutes. The process is faster and more reliable and that translates to less context-switching and an increased ability to focus on a single task."

— Director of Engineering at an **European online financial management platform**

Questions to ask:

- What top-line revenue can be created by removing the wait time?
- How long is your team "waiting" for some activity to finish?
- How much time is your team spending on manual tasks that cause employee dissatisfaction?
- How much time is spent waiting for environments to be created by other teams?
- How much can your bottomline improve with increased productivity of your team?
- How is your team collaborating with other parts of your organization?

"Collaboration has also improved thanks to GitLab. Not only is the team not having to spend time dealing with disparate tools, but collaboration is easier. We are using one repository, not only together with several teams, but even together with several business units. There are a lot of changes that impact a lot of teams and GitLab really makes it easy for us to collaborate and get approval from all the teams that a change is accepted"

— Product Manager at **a German automotive technology supplier**

Questions to ask:

- How much additional revenue can be earned if you release faster?
- How often do you deliver software compared to your competitors?
- How much time has your team spent in developing features that don't match customer requirements?
- How fast are you able to react to competitive shifts and customer needs?
- How much does improved velocity increase your chances of retaining customers?

"If you want to speed up the delivery cycle, you need to simplify your ecosystem. And we've been doing that with GitLab along the way, it's critical for developers to have one single point of contact and one simple inter-face to increase the speed of delivery."

— VP of Cloud Operations **a large technology company**

Gains in velocity

Is your team delivering software once a month? Once a quarter? Once every 6 months? How does this pace of software delivery compare to your competitors? Not only is it important to deliver software in smaller chunks, to unleash its value faster, but also automate the entire process to minimize availability and performance issues in production.

Increasing the velocity of software delivery helps you get new features to customers — and receive their feedback — faster. That means your team can learn and pivot faster to provide a superior user experience, and open up opportunities for additional revenue generation.

Gains in availability and performance

Detecting errors early in the development process is much cheaper and less time consuming than detecting them in production. **National Institute of Standards and Technology (NIST)** projected that the cost of detecting and fixing bugs increases exponentially with the amount of time the bug spends in the SDLC. Fixing in production is the most expensive and most risky. Automating your software delivery process helps to detect these errors early — thereby increasing the availability and performance of your software in production. Improving your availability by just 0.1% equates to reducing your downtime by over 500 minutes per year.

Questions to ask:

- How many minutes of downtime does your organization have per month, per year?
- How much does every minute of downtime cost your organization?
- How many defects are detected in the design phase vs. development phase vs. testing phase vs. staging phase vs. production?
- How much time is wasted by a developer in context switching to cater to urgent or emergency issues found in production?

> "Detecting bugs earlier in the life cycle has increased output capabilities. Quality testing goes smoothly right before release so that release dates can be met and marketing can be executed as planned."
>
> — Principal Engineer at a **Japanese cloud services provider**

> "Engineers are able to detect bugs earlier in the software lifecycle because they're deploying smaller releases, faster. "We're detecting bugs very, very fast and they're also fixed very fast. They can be fixed in maybe 10 minutes to one hour ...and also automatically fixed in the production environment. This is a great value."
>
> — DevOps Engineer at a **Swiss IT partner** for leading financial and insurance service providers.

Cost of learning

Cost of learning and ease-of-use is an intangible benefit that is often overlooked. The reality is DevOps solutions are used not just by developers but by a cross section of employees within an organization, like operations, site reliability engineering (SRE), security teams, and so on.

If you have multiple tools to automate your software delivery, it means your employees have that many more tools to learn and use on a daily basis — and that means your team is spending time on non-business-value generating activities. The easier the solution is to learn and use, the greater your productivity savings.

Questions to ask:

- How much time and money is the organization spending on getting team members up to speed?
- How many tools does a team member need to learn to get their job done?
- Is the learning curve of different tools blocking collaboration between various teams?
- How quickly is a new team member able to start contributing actively?

> "The team has observed that GitLab has helped with onboarding new hires. By only having to learn one tool, with useful templates, new hires have been able to push code on their second day."
>
> — Head of DevOps Enablement at **a large bank in Australia**

Cost of maintenance

Yes, best of breed solutions could minimize vendor risk. But having a specialized tool for every part of your SDLC is difficult to set up and maintain, and it also drains your team's resources.

Imagine the scenario where one vendor upgrades their software and that software is integrated with the five other pieces of software you use to automate your software delivery. That means every time there is an upgrade, all those integrations need to be retested. If there are changes, those need to be reintegrated. This affects reliability too. If your DevOps environment is offline, consider the cost of idle developers.

With more vendors updating software more frequently, the maintenance overhead for one integration quickly adds up and can cause efficiency reductions. To enable your team members to focus on their core work, your company needs tools that are easy to maintain and have integrations tested out of the box.

Questions to ask:

- How many integrations are your teams managing?
- How many hours does your team spend on maintaining these integrations?
- How many people do you have on your team to manage your DevOps toolchain?
- How many hours are spent managing licenses used across the various tools when a new project is initiated?

> "With developers using multiple tools, there was little visibility into the status of projects. The maintenance and integration required to keep the developers up and running was taking up too much time and slowed throughput. We want to use as few tools as possible, and we want to stay on the same tool that everyone can use. With GitLab, my maintenance became easy because I had to maintain fewer servers. And I didn't have to maintain the connections."
>
> — Configuration Manager at a **robotic equipment manufacturer in the Netherlands**

Questions to ask:

- Does the software provide all the capabilities you need or would you require additional tools?
- How much upfront and ongoing cost on a monthly basis do you need to budget for?
- Are there any variable costs to the software (e.g., usage-based costs that will be higher for your use case)?

> “We were maintaining a total of three internal tools for the course of several years. It was a full-time job for an engineer to manage and maintain the tools, which reduced time for software engineering. On top of that, having data and user permissions in various places caused a lot of context switching, which was time-consuming and inefficient. With GitLab, we are able to truly focus on deliverables instead of performing updates and toolchain maintenance.”
>
> — Chief Technology Officer at **an engineering consulting firm**

Cost of acquisition

License costs are the most obvious costs that many organizations consider when trying to calculate ROI. A low cost of acquisition does not necessarily mean a better ROI, though. It is important to evaluate the cost of owning and maintaining the software over time.

Start free trial

Start your 30-day GitLab free trial

Why GitLab for Automated Software Delivery?

With **Automated Software Delivery with GitLab**, your organization can:

- **Scale your SDLC for cloud native adoption**
  Eliminate click-ops, introduce controls essential for cloud native adoption
- **Make every change releasable**
  More testing, errors detected earlier, less risk
- **Improve the developer experience**
  Minimize repetitive tasks, focus on value-generating tasks

You can do all of this with one single license, one permission model, one interface, and one data model to deploy software to multiple clouds — giving your team the flexibility to focus on creating business value instead of managing a complex toolchain.

Security and compliance on your mind? Integrating security and compliance into your DevOps lifecycle is easy with GitLab. Learn more about **DevSecOps with GitLab**.

Get free trial

Start your 30-day GitLab free trial