GitLab

Automated Software Delivery

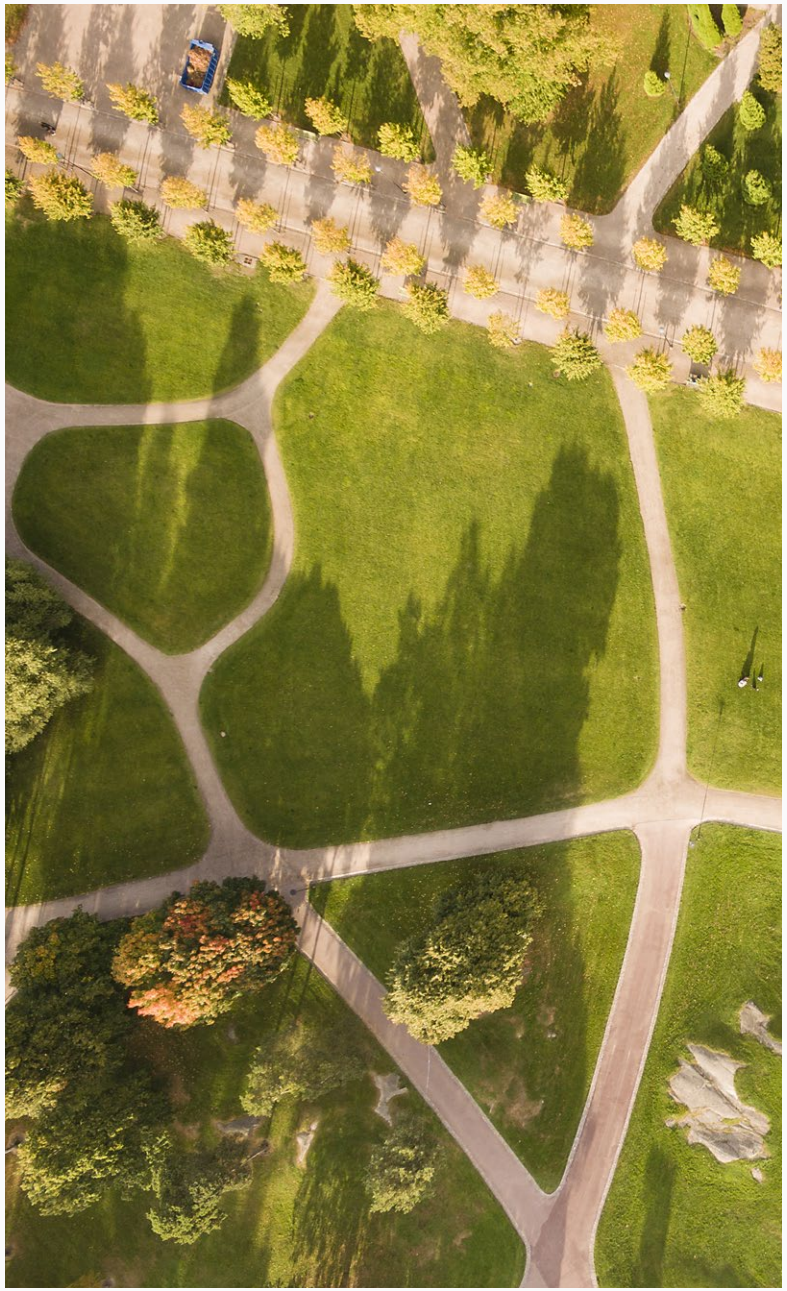
A 3-phase approach to modernizing your existing software delivery process

What’s inside?

A 3-phase approach for automating software delivery

Modernizing your existing software delivery processes does not take place from one day to the next; rather it is a journey, where improvements can be made in a phased manner. Organizations are unique and their adoption of DevOps principles can take many forms and can follow different paths that are appropriate to them.

No matter where you are in your adoption journey, automating your software delivery can improve safety of your development lifecycle and releases and lower risk through consistent and repeatable processes. We suggest a three-phase incremental approach to automating your software delivery.



Phase 1: Consolidate DevOps tools and start with the basics

For organizations that are beginning to embrace DevOps practices, a good first step is to lay down the foundation on top of which your software delivery automation will be built. Ideally, this foundation should be encapsulated in a platform with a single data model that can support all DevOps stages, from planning to multi-cloud deployments. Some potential benefits of a single data model are better correlation of data across all stages, better data harvesting and visualization leading to the creation of higher-level business value assets, like reports and dashboards that can provide insight into your teams' software delivery and operational performance and efficiency.

With this in mind, put together a plan to consolidate and rationalize your heterogeneous DevOps tools into a platform that supports a single data model. Consolidating your DevOps tools does not necessarily mean you have to immediately relinquish your other tools right from the start. Since you are beginning to embrace DevOps practices, here are some good starting points:

- **Move everything to code.** This helps to make your processes repeatable, scalable, and less error prone. Most version control and collaboration (VCC) tools that support DevOps processes use git as their underlying technology. So, it should be fairly straightforward to migrate from a best-of-breed git-based tool to the single platform. However, if you'd like to keep your VCC tool, you could import or mirror the VCC-related data from it to the single platform.

- **Use issues, merge/pull requests, epics, iterations/sprints, and milestones.** Issues, where product problems or new features are described, and merge requests, where solutions are developed, are key inputs to the release planning process. As components of the release, issues and MRs provide the auditability and tracking of application changes done by the collaboration of DevOps Engineers, System Administrators, and Developers. To track groups of issues with the same theme, you can create epics.

> **"GitLab is the one tool that connects our whole team. You always see GitLab open and everything is based on GitLab. GitLab is the backbone of our software development."**
>
> **— Head of Software Development from a global IT service provider**

Milestones are a way to track issues and merge requests created to achieve a broader goal in a certain period of time. They allow you to organize issues and merge requests into a cohesive group, with an optional start date and an optional due date. Iterations are a way to track issues over a period of time and help you track velocity and volatility metrics.

Iterations can be used with milestones for tracking over different time periods. You can track the sprints through their detailed pages, which include many progress metrics. As you assemble epics, milestones, and iterations, you can visually track the release progress via the Roadmaps page, which helps to streamline the development lifecycle and release processes.

- **Set-up approval gates for protecting your production environment.** As part of the release approval gates, you can protect the production environment by specifying who is allowed to deploy to it. Specific role and responsibility assignments streamline the approval gates and release process.

- **Leverage the built-in capabilities for the automation of the release creation process.** If the release evidence is created by the built-in release creation process, then take advantage of this automation. The release evidence contains data related to the release, such as milestones, issues and job artifacts included in the last pipeline that ran, among others. This streamlined process helps you reduce the release cycle times.

- **Set-up deploy freezes.** You can set a deploy freeze window to temporarily halt automated deployments to production. This prevents unintended production releases during a period of time to help reduce uncertainty and risk of unscheduled outages.

> "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 interface to increase the speed of delivery."
>
> — VP Cloud Operations from an **independent API gateway vendor**

Phase 2: Implement continuous delivery

As you continue adopting more DevOps practices within your organization, a good second step is to set up continuous delivery. In this second phase, every change is automatically deployed to the user acceptance testing (UAT) or staging environment with a manual deployment to production. In this scenario, there is no need for a deploy freeze, you can cut a release from staging at any point in time.

To begin the adoption of continuous delivery in this second phase:

Leverage any built-in CI/CD pipeline templates included in the platform. These templates provide many benefits, such as:

- **Automatically creating CI/CD pipelines** and reduce time spent manually creating your own, so your developers can spend more time delivering value to the business. You can even start with a template and modify it to fit your needs.
- **Automatically standing-up the staging and production environments**, deploying the application to them, and putting advanced deployment techniques at your disposal. This includes incremental and canary rollouts, increasing your development cycles' productivity and speed, accelerating your time-to-market. In addition, incremental rollouts lower the risk of production outages delivering a better user experience and customer satisfaction. Advanced deployment techniques, like canary, incremental, and blue-green also improve development and delivery efficiency streamlining the release process.
- **Building template jobs** which can automatically detect the languages of your application and apply the appropriate build strategy to create a Docker or container image of the application and store it in the platform's built-in container registry. Faster and more reliable releases happen when you have build components, like container images, that are readily available to the development lifecycle and release processes in a uniform and consistent manner. You could also leverage the platform's built-in package registry, which supports many packaging technologies (e.g. maven, npm, etc.)

In short, all these templates can help you streamline your software delivery processes.

Start using any review pipeline templates included in the platform. These pipelines allow stakeholders to visualize what specific features will go into production. As updates are made to the application via merge/pull requests, they kick off review pipelines, which streamline the review process including the automatic creation and destruction of an ephemeral review environment, on which stakeholders can collaborate and verify the updates to the application before they are merged to the main branch. These

review pipelines can help increase code quality, reducing the risk of unexpected production outages.

Take advantage of any built-in rollback and auto-rollback capabilities. All updates to the environments, managed by the platform, are tracked and kept as an auditable list of changes. If a rollout puts your production environment in an unstable state, you can quickly roll production back to a previous working state, with the click of a button, by choosing it from the auditable list. Rollbacks can speed up recovery of production in case of failures lowering outage times and leading to better customer satisfaction and user experience.

Leverage any pipeline scheduling capabilities included in the platform. Pipelines usually run automatically. However, if you would like to schedule the execution of a pipeline once a day at midnight, for example, so that the user acceptance testing or staging environment can have the most recent version of the application on a daily basis, make sure to use any built-in scheduling capabilities. Scheduling pipelines can improve the efficiency of the development lifecycle and release processes.

Monitor how your applications are performing in production via built-in dashboards available in the platform. By tracking applications performance, you can quickly identify and troubleshoot any production issues. There are a few ways you can do this:

- **Monitor a specific environment to track system and application metrics**, such as system and pod memory usage, and # of cores used. Through this monitoring, you should be able to track markers when updates were introduced to the environment, so that fluctuations in the metrics can be correlated to a specific update. Monitoring reduces the time to identify, resolve and preempt production problems lowering the risk of unscheduled outages. It also provides an opportunity to do business activity monitoring and optimize cloud costs. Not only is this type of monitoring useful to release managers but also to DevOps Engineers, Application Operators and Platform Engineers.

- **Create alerts to detect out-of-range metrics**, which you should be able to visualize on overall operations metrics dashboards as well as on specific environment windows. Alerts can also automatically trigger ChatOps and email messages to appropriate individuals or groups. You should be able to manage alerts from a centralized dashboard, a single location from which you can assess and handle alerts, which may include the manual or automatic rollback of a release.

- **Track and monitor your development lifecycles and release progress through value stream analytics dashboards**, where you can check your project or group statistics over time and see how your team improves in the number of new issues, commits, deploys

and deployment frequency. Value stream analytics is useful in order to quickly determine the velocity of a given project and it also points to bottlenecks in the development process, enabling management to uncover, triage, and identify the root cause of slowdowns in the software development lifecycle.

- **Track and monitor your releases through pipeline analytics dashboards.** Pipeline analytics shows the history of your pipeline successes and failures, as well as how long each pipeline ran to help you understand the health of your projects and their continuous delivery. They should include your DORA4 metrics, which are performance metrics that measure the effectiveness of an organization's development and delivery practices.

Track overall operations via consolidated built-in dashboards available in the platform. For example, through an operations dashboard, you get a summary of each project's operational health, including pipeline and alert status. Similar to an operations dashboard, an environments dashboard can provide a cross-project environment-based view that lets you see the big picture of what is going on in each environment. All these dashboards provide you with the operations insights that you need to understand how your development cycles and releases are performing in production and quickly identify and troubleshoot any production issues.

Start identifying and planning GitOps processes that can support your software delivery. The application of DevOps principles and best practices to the management and automation of your infrastructure is what GitOps is all about. Start identifying what infrastructure components can be managed and automated using infrastructure-as-code (IaC) in git. Whether these are physical, virtual or containerized, start planning what GitOps flows would make sense to establish to ensure that your infrastructure remains in sync with the IaC files that describe it.

Phase 3: Implement continuous deployment

As you mature in your adoption of DevOps best practices, the third step is to get set-up for continuous deployment. In this third phase, every change is automatically deployed to production.

To start the adoption of continuous deployment in this third phase:

- **Take advantage of any built-in capabilities that can automatically handle continuous deployments.** This will save you time and can increase productivity and speed of development and deployment to production, accelerating your time-to-market.
- **Start using progressive delivery/developer experimentation techniques.** Think of progressive delivery/developer experimentation as continuous delivery with fine-grained control over how and who gets to experience the updates to your application. Feature flags are a good example of this. If you would like to introduce a feature to a segment of your end users in a controlled manner in production, you can create Feature Flags. Feature flags help you reduce risk, allowing you to do controlled testing, and separate feature delivery from customer launch.

30%
improvement in developer productivity

2.6%
reduction in build times

50%
faster to launch a new application

- **Leverage any built-in audit events dashboards.** Audit events dashboards track important events such as who performed certain actions and the time they happened. For example, these actions could be a change to a user permission level, who added/removed a user, who introduced a feature flag, for example.

- **Start using any security dashboard included in the platform.** You can check security and compliance related items of your projects by leveraging the built-in Security dashboards in the platform. The audit events and security dashboards can help you preempt out-of-compliance scenarios to avoid penalties. They can also streamline audits, provide an opportunity to optimize cost, and lower risk of unscheduled production outages.

- **Start implementing GitOps processes that support the automation of your software delivery.**

"GitLab has really helped us because we can work with a lot of people on projects, maintain the quality, make sure that everybody checks each other's work, and then deploy it on an infrastructure that doesn't bear any surprises because it just follows through the continuous deployment. It just follows wherever the project is going."

— CTO from a large data science and engineering firm in Europe

Begin your journey to DevOps modernization

No matter where you are in your adoption journey, automating your software delivery can improve safety of your development lifecycle and releases and lower risk through consistent and repeatable processes. Modernizing your existing software delivery processes doesn't take place overnight, rather it is a journey, where improvements made in phases get you to the destination, and GitLab will help you get there.

Contact Us

Learn more >