HOW TO ACHIEVE DEVSECOPS WITH GITLAB CI/CD

Shift left with built-in security tools and best practices

GitLab

Table of Contents

Introduction

DevOps practices have enabled software developers (devs) and operations (ops) teams to accelerate delivery through automation, collaboration, smarter feedback loops, and iteration. DevOps goes beyond the Agile and lean practices it came from to make faster software delivery possible. DevOps represents a cultural shift that focuses on not only speed, but efficiency as well.

While DevOps has removed many of the barriers to faster delivery, one area of development has sometimes felt like an outsider looking in: security.

Traditional application security is usually a final step of the development lifecycle. Dedicated security professionals run tests to identify vulnerabilities, prioritize them by risk, and then triage for remediation. Looking at security as an afterthought in the development process doesn't breed optimal results and causes bigger problems in the long run. With security acting as a gatekeeper for deployment, it can feel like a roadblock to innovation by developers and engineers, rather than an equal partner.

With security and compliance being so important, especially with **the security threats** out there today, teams are realizing that it's not enough to just practice DevOps with security thrown in at the end. Balancing business velocity with security is possible. For teams wanting to innovate, going beyond traditional DevOps will not only be important, it will be essential.

What is DevSecOps?

DevSecOps brings security into the development lifecycle by integrating security processes and actions into the same pipelines developers and operations are already using. This shifts security earlier into the development workflow in a process known as shifting left – but it's not just about making developers responsible for security. In DevSecOps, teams rely on tools and processes that treat security with the same scale and rapid feedback loops enjoyed by modern software development.

In the past, security tools were highly specialized and not necessarily developer-friendly. Developers wanted command-line tools that could be automated, customized for various configurations, and imported into bug trackers. But traditional security scanners were made for security teams and CISOs, whose goals are governance, security policy compliance, and risk management – not necessarily development velocity.

Over time, security vendors and developer platforms have adapted their products to address the needs of both: Analytics and reports needed by CISOs with integrated workflows needed by developers. CI/CD pipelines can incorporate security testing either natively or as an integration from a third-party vendor. This has allowed security to shift left while also being robust enough to identify vulnerabilities. Teams are more poised than ever to adopt a real DevSecOps culture.

CI/CD: The key to DevSecOps

DevSecOps integrates security controls and best practices into the DevOps workflow through CI/CD pipelines. As more teams try to shift left, automated security testing streamlines adoption and scalability. In GitLab's 2020 DevSecOps Survey, a respondent summarized the importance of testing and continuous integration:

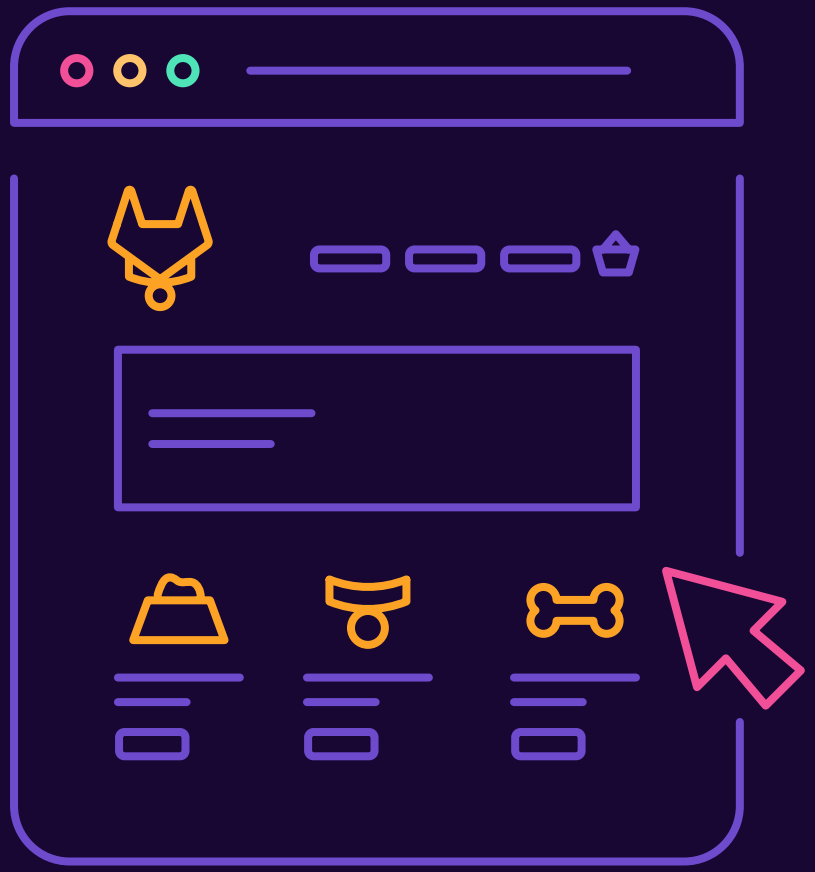
> **"Automated testing and continuous integration have made our deployments safer and more optimized. Now everyone in the team has the permission to deploy the code."**

To achieve DevSecOps, a robust CI/CD strategy with built-in security features will be a main component. GitLab's open DevOps platform with built-in CI/CD brings these requirements into the development lifecycle naturally. Teams that adopt a good CI/CD strategy are not only able to develop better, faster software, they also improve business outcomes, identify bugs, and catch vulnerabilities before they ever reach users.

DevSecOps in action:
Tanuki Pets

The benefits of DevSecOps really shine when a diverse team needs to collaborate on an urgent project. Sound familiar? In this example, we created a company that needs to innovate quickly to avoid losing customers to a competitor.

Tanuki Pets is a fictional company that has an online website and mobile app that sells pet supplies, pet food, and other pet products. Tanuki Pets is a Java application hosted in an on-premise data center.



The Tanuki Pets Team

Rachel is the configuration manager.
They are responsible for the version control tool and repository, manages manual and automatic deployment of environments, and develops best practices and processes in the area of coding and code management.

Delaney is the development lead.
They started developing the company website four years ago. Delaney still maintains the code and last year got promoted to a team lead.

Sasha is a new developer.
They just joined Tanuki Pets as a developer and report to Delaney. Sasha joined Tanuki Pets because they like pets, they have a dog and two cats at home, and they thought it would be fun to develop a website for pets. They love open source.

Parker is the product manager.
They work closely with Presley, the product designer, to define new functionality that will increase the company business. Parker tries to push for releasing features to production fast – but this usually doesn't happen.

Simone is the QA engineer.
Their responsibility is to ensure that the team releases high quality products that are usable and with a fun user experience. Most tests are manual.

Presley is the product designer.
They are creative and come up with great ideas and solutions. Presley's job is to translate the product's mission into an effective, empathetic, and efficient user experience.

PetBuy.com, a Tanuki Pets competitor, just released their own mobile application with a cool feature that enables users to upload their pet photos and rate other pet photos to win prizes and discounts. Pets that receive more than 200 votes trigger a **70%** off discount that their owners can use for a future online purchase.

In addition to using this feature to promote their new mobile app, users are enjoying interacting with pets all over the world and interacting with the brand.

Unfortunately, for our friends at Tanuki Pets, sales drop **30%** practically overnight. Mark Zhang, Tanuki Pets CEO, feels the team must deliver new social features and special promotions as soon as possible to compete with PetBuy.com.

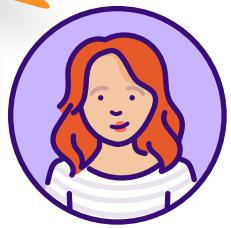
> **The goal:** **Deliver new features and innovations so that Tanuki Pets can better compete against PetBuy.com.**
>
> **The alternative:** **Without rapid action to improve and mitigate the decline in sales due to the competition, Tanuki Pets is unlikely to survive long-term.**

All the Tanuki Pets team members need to collaborate in order to deliver features as quickly as possible. But everyone on the team has different incentives, priorities, and dependencies. Collaboration is doable, but not easy, and the Tanuki Pets CEO asks that all team members give this new project their highest priority.

We need to release a new module that enables customers to upload their pets photos and react to others' pet photos. How long is this going to take to release?

It will take me at least a week to design this if I work as fast as I can...

I need at least two weeks to write the code, but I'll also need Rachel to provision a dev environment for me to use but that could take days to get approved...



IMPROVING THE PROCESS

As is the case for most businesses, the pace of innovation needs to be greater than or equal to competitors to outpace them and, ultimately, succeed. It's sometimes forgotten that engineering goals can have a direct impact on business outcomes: The faster that features can be released and enjoyed by users, the sooner businesses can generate revenue from that code. The Tanuki Pets team realizes they must find a way to work faster and deliver quality results, safely. 10 weeks from start to finish is just too long to wait.

The reality is that there rarely is a perfect time to improve processes. The team knows they need to adopt a DevSecOps methodology if they have any hope of releasing their own pet photo feature.

> **Sasha:** "We need to adopt DevSecOps and see if it generates the results we need. The data is there and shows it can dramatically improve the velocity and quality of software produced by our team."
>
> **Sasha:** "What about the manual tasks that eat a lot of time, like deploying a test environment and testing the code? If we could just automate these things and have automatic security scans, it could easily save us ten days or more."
>
> **Delaney:** "Honestly, I can barely keep up with the processes we have today."

When making the case for CI/CD, or any new technology, IT leaders need to be convinced that adopting new tools or processes will be worth it in the long run. Shifting to DevSecOps requires an investment in time and resources that can sometimes take years.

In this scenario, Sasha recommends GitLab as a unique solution for the team's needs.

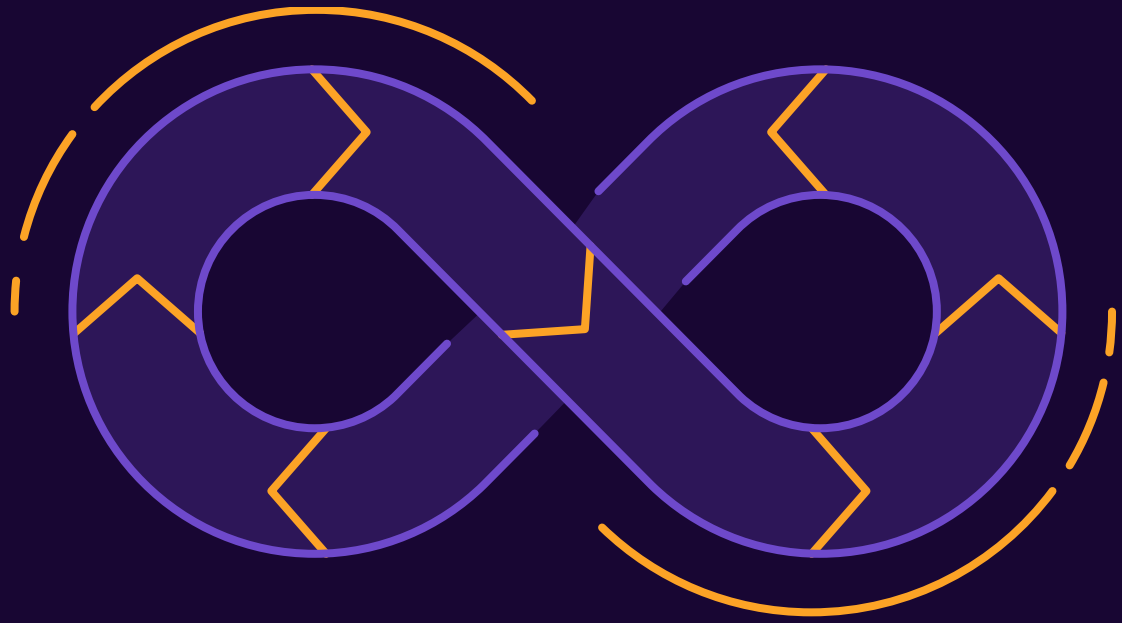
GitLab is an open DevOps platform, delivered as a single application, including built-in CI/CD. As a single application, it has all components the team needs in one place: single UI, single data storage, and single vendor.

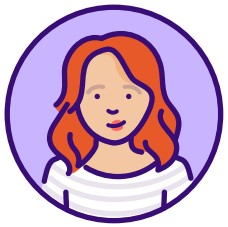
Getting started with GitLab CI/CD

Since GitLab is available via SaaS or self-managed versions, rather than install on their on-premise data center, the Tanuki Pets team opts to use GitLab SaaS so they can start using it right away.

The team signs up for the GitLab free trial which gives them access to all GitLab Ultimate features they'll need for this project, including:

- Agile project management
- Source code management
- Continuous integration and delivery
- Application security testing
- Infrastructure-as-code
- Logging, monitoring, and tracking



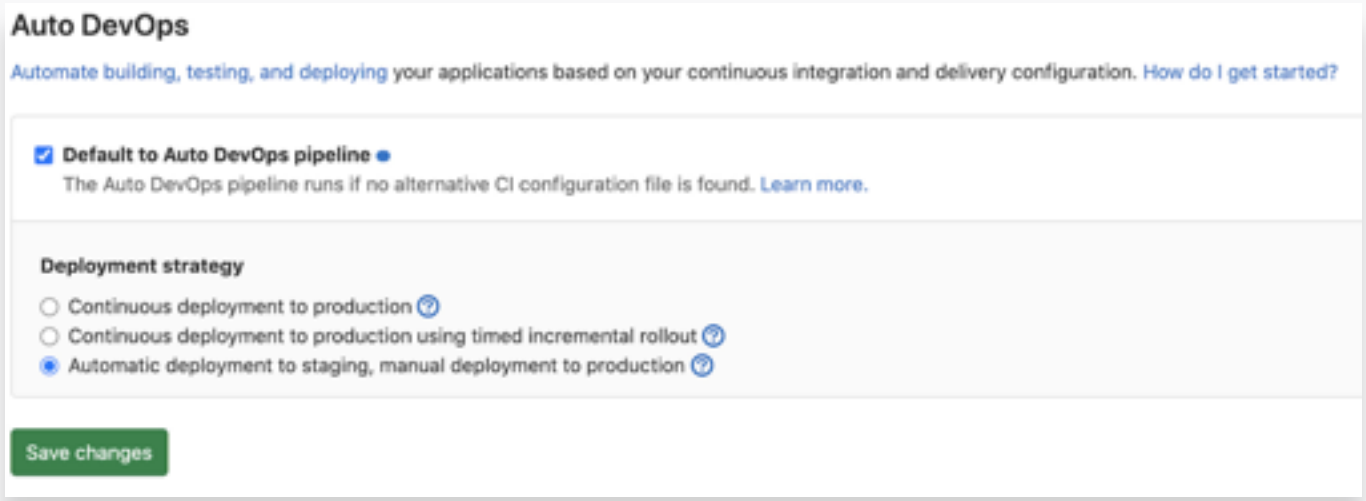


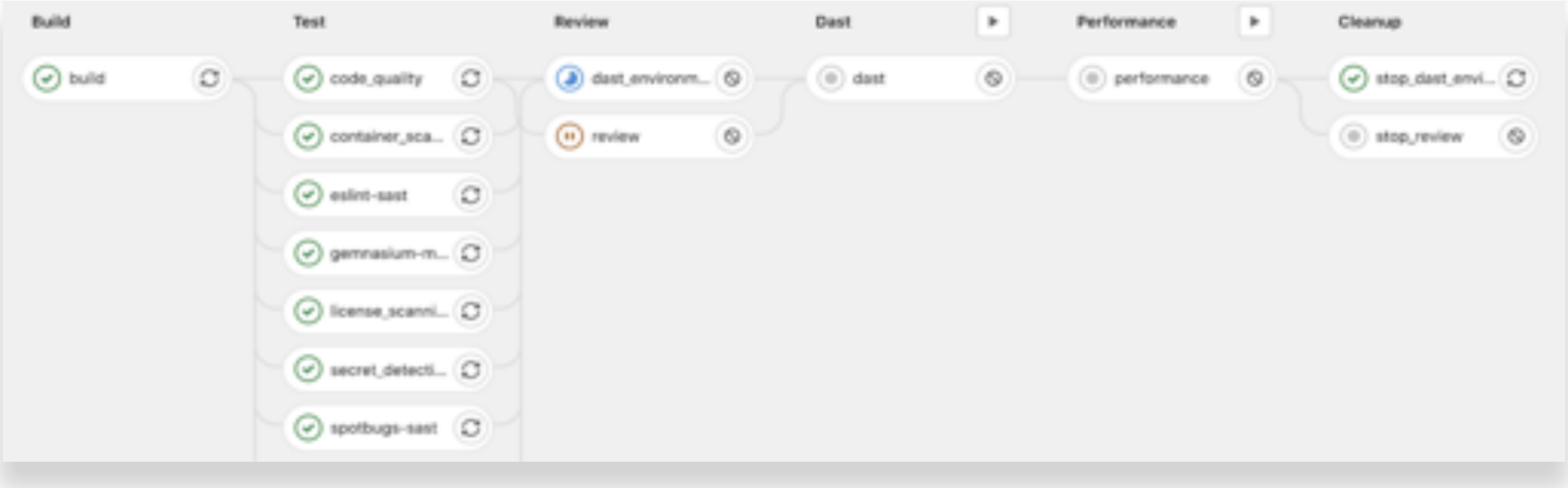
I just created a group for our team and added all of you. Then I created a project in our group and pushed Tanuki-Pets source code to it in less than two minutes. Now all of us can clone the code from the GitLab repository and start developing locally.

GitLab makes implementing and configuring CI easier out-of-the box with Auto DevOps. Rather than building a YAML file from scratch, or trying to configure existing scripts, Auto DevOps provides predefined CI/CD configurations that automatically detect, build, test, deploy, and monitor applications. Since it's enabled by default, all the Tanuki Pets team needs to do is start a pipeline job and Auto DevOps does the rest. Everytime there's a code change, GitLab will trigger a CI/CD pipeline to build and test the code.

In five minutes, we have a fully functioning DevSecOps environment running our first pipeline. Even better news: Our code and the build passed without any problems!

Auto DevOps settings





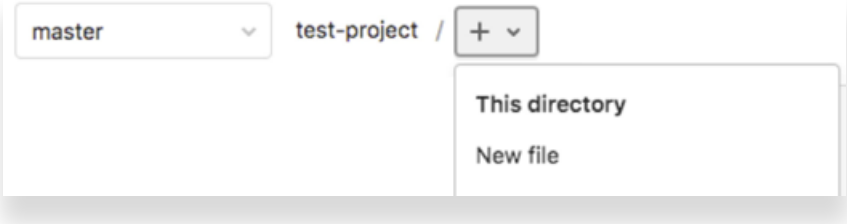
Configuring continuous integration

While Auto DevOps makes it easy to get started right away, GitLab CI/CD can be configured for any need. GitLab uses pipeline as code and a `.gitlab-ci.yml` file saved in the root of the repository. GitLab will detect the syntax automatically and run the steps defined once new code is pushed.

Creating a new `.gitlab-ci.yml` file is an easy process.

1. **Go to Project overview > Details.**

2. **Above the file list, select the branch you want to commit to, click the plus icon, then select New file:**



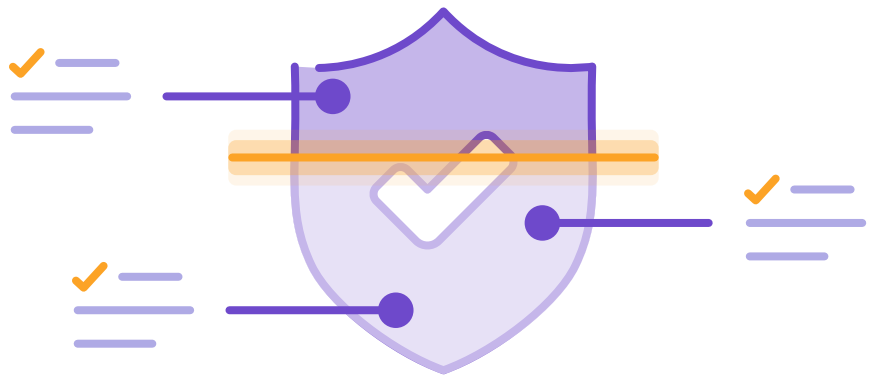
3. **For the Filename, type `.gitlab-ci.yml` and in the larger window, paste this sample code:**

```
build-job:
  stage: build
  script:
    - echo "Hello, $GITLAB_USER_LOGIN!"

test-job1:
  stage: test
  script:
    - echo "This job tests something"

test-job2:
  stage: test
  script:
    - echo "This job tests something, but takes
more time than test-job1."
    - echo "After the echo commands complete,
it runs the sleep command for 20 seconds"
    - echo "which simulates a test that runs 20
seconds longer than test-job1"
    - sleep 20

deploy-prod:
  stage: deploy
  script:
    - echo "This job deploys something from the
$CI_COMMIT_BRANCH branch."
```

GitLab provides quick start security scan templates, so embedding security into a CI/CD pipeline is a snap. Get started quickly with Dependency Scanning, License Scanning, Static Application Security Testing (SAST), and Secrets Detection by adding the following to your `.gitlab-ci.yml`:

```
include:
  - template: Security/Dependency-Scanning.gitlab-ci.yml
  - template: Security/License-Scanning.gitlab-ci.yml
  - template: Security/SAST.gitlab-ci.yml
  - template: Security/Secret-Detection.gitlab-ci.yml
```

This is by no means an exhaustive list. GitLab has even more scans available that can be added at any time, depending on your needs.

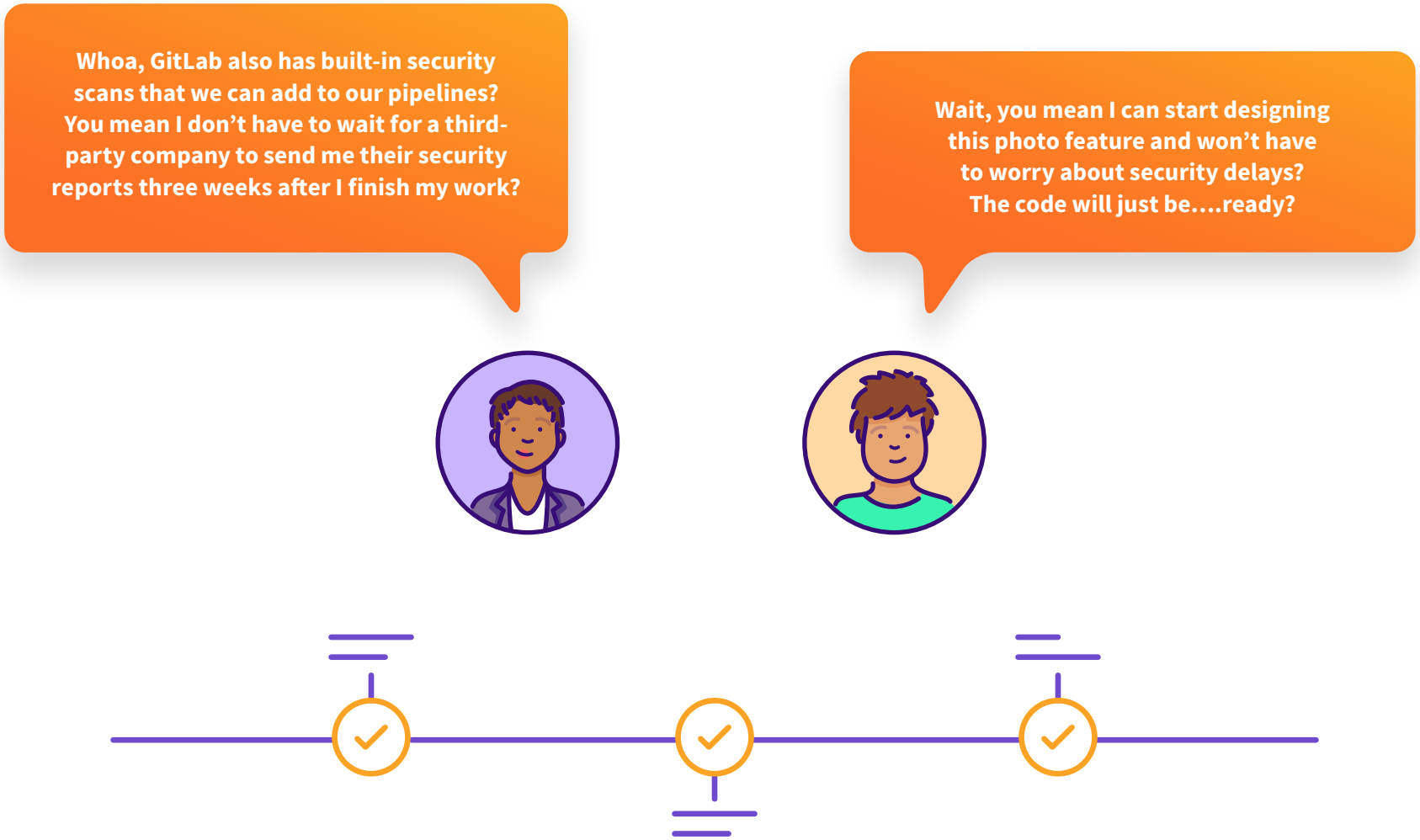


Security scanning tools

GitLab uses the following tools to scan and report known vulnerabilities found in your project.

Secure scanning tool	Description
Container Scanning	Scan Docker containers for known vulnerabilities.
Dependency List	View your project's dependencies and their known vulnerabilities.
Dependency Scanning	Analyze your dependencies for known vulnerabilities.
Dynamic Application Security Testing (DAST)	Analyze running web applications for known vulnerabilities.
API fuzzing	Find unknown bugs and vulnerabilities in web APIs with fuzzing.
Secret Detection	Analyze Git history for leaked secrets.
Security Dashboard	View vulnerabilities in all your projects and groups.
Static Application Security Testing (SAST)	Analyze source code for known vulnerabilities.
Coverage fuzzing	Find unknown bugs and vulnerabilities with coverage-guided fuzzing.

GitLab also provides support for end-to-end testing with Selenium and WebdriverIO. While unit tests are used the majority of the time, end-to-end testing, also called broad-stack or full-stack testing, is extremely valuable for large applications so that you can know that the deployment went as intended, that your infrastructure is up and running, and that all of your code works well together.



DEPLOY TEST ENVIRONMENT

I usually spend at least two days configuring and deploying test environments. With GitLab CI/CD, can I automate this process?

The easiest way to make this happen will be to build a Docker image from the code and deploy it using Kubernetes. GitLab has a robust Kubernetes integration and provides cloud-agnostic support for cloud native development so that you can deploy anything you want, anywhere, anytime.

You can use a predefined template for building the docker image and a CI job template to orchestrate the deployment.

To create a Docker image and push it to GitLab container registry, include this template in your CI config file:
`Jobs/Build.gitlab-ci.yml`

To add a deploy job: add deploy stage, and paste this job template.

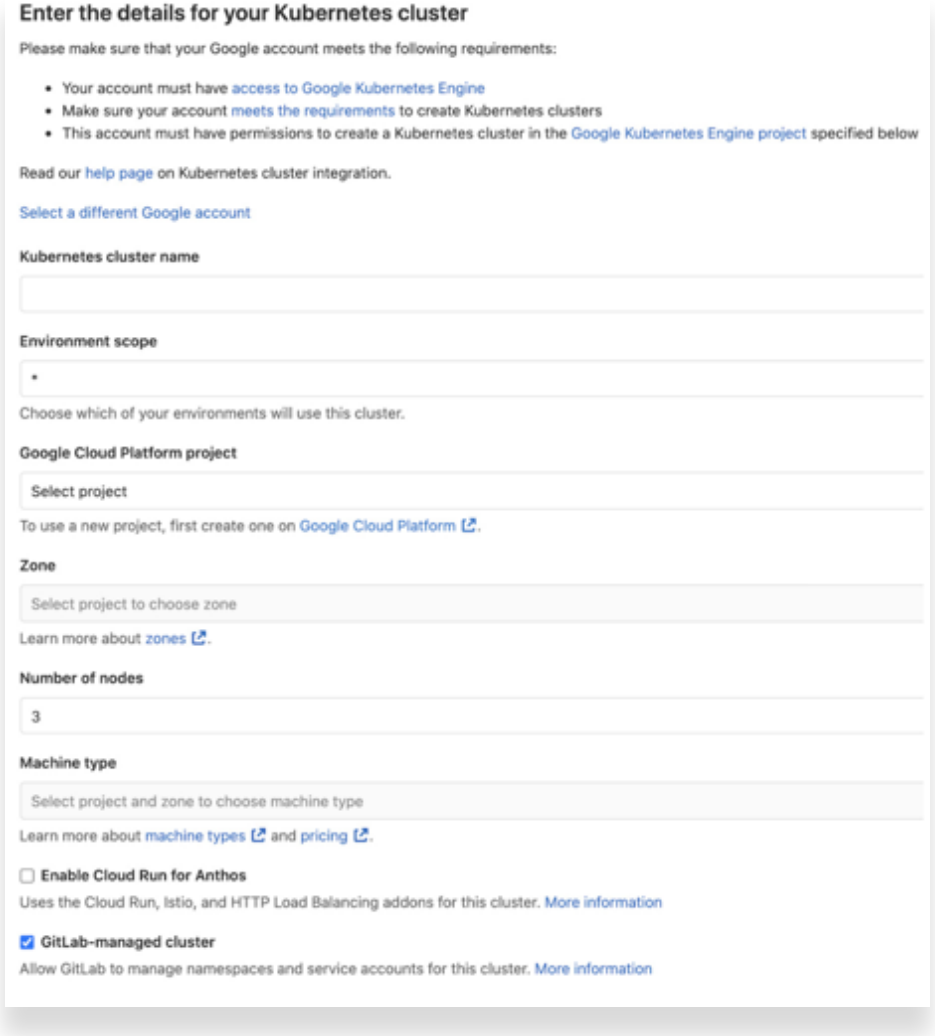
```
deploy-test:
  image: "registry.gitlab.com/gitlab-org/cluster-
integration/auto-deploy-image:v1.0.7"
  stage: deploy
  script:
    - auto-deploy check_kube_domain
    - auto-deploy download_chart
    - auto-deploy ensure_namespace
    - auto-deploy initialize_tiller
    - auto-deploy create_secret
    - auto-deploy deploy
    - auto-deploy persist_environment_url
  environment:
    name: deploy/$CI_COMMIT_REF_NAME
    url: http://$CI_PROJECT_ID-$CI_ENVIRONMENT_
SLUG.$KUBE_INGRESS_BASE_DOMAIN
  artifacts:
    paths: [environment_url.txt, tiller.log]
    when: always
  rules:
    - if: '$CI_KUBERNETES_ACTIVE == null || $CI_
KUBERNETES_ACTIVE == ""'
      when: never
    - if: '$CI_COMMIT_BRANCH == "master"'
      when: never
    - if: '$REVIEW_DISABLED'
      when: never
    - if: '$CI_COMMIT_TAG || $CI_COMMIT_BRANCH'
```

CREATE A KUBERNETES CLUSTER

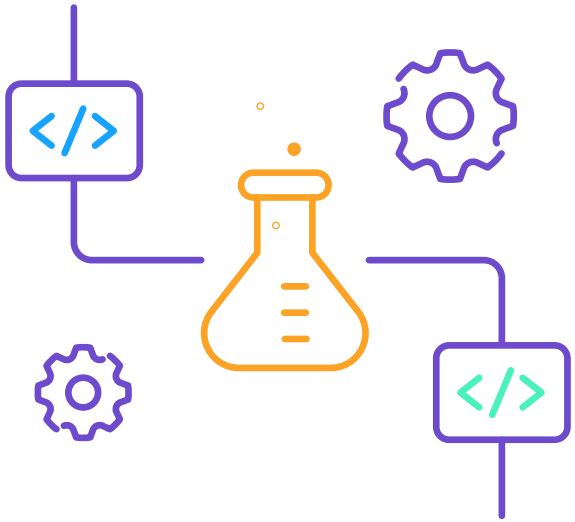
GitLab provides a wizard to create clusters and install on them all necessary applications. To start the wizard, go to the overview page of your project and click on ‘Add Kubernetes cluster,’ then follow the instructions.

Once the cluster is created and connected to your project, you can deploy your app to it.

For each commit we push to the server, GitLab will create a Docker image, publish to the container registry, and deploy it in Kubernetes!



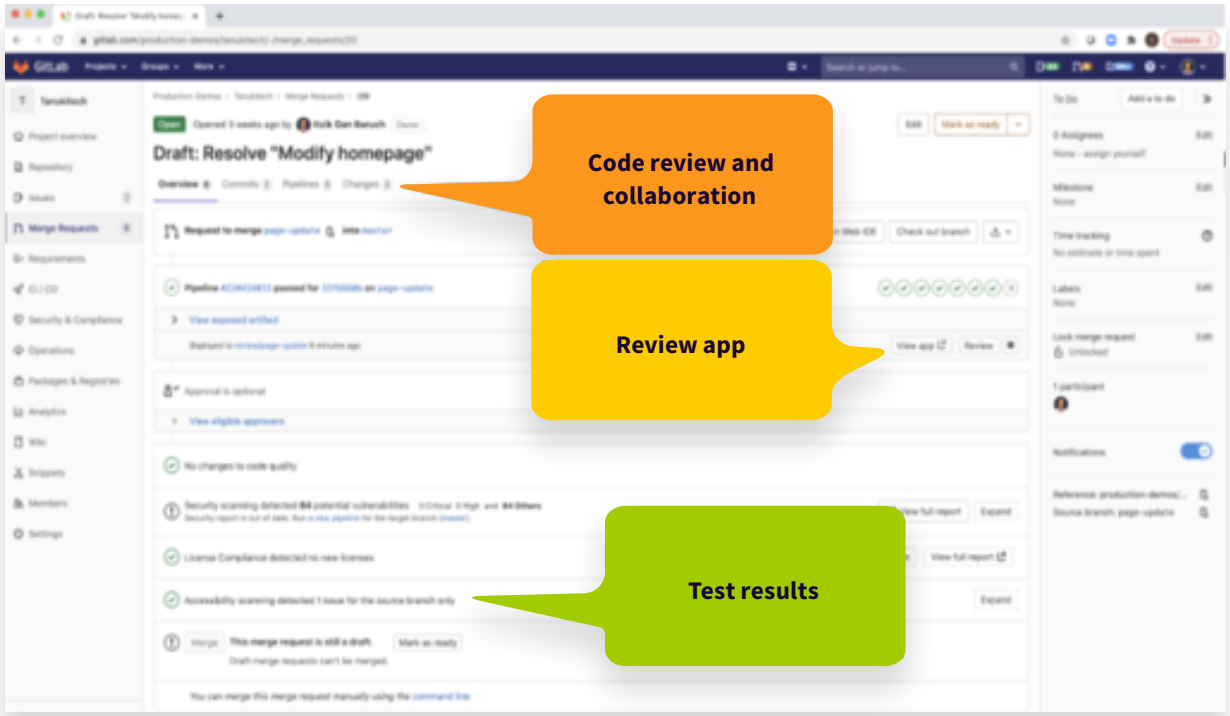
This is awesome, but where do developers see reports and results?



SEEING THE TEST RESULTS

In GitLab, the test results are available from the merge (pull) request (MR). MRs include build and test results, other valuable insights such as code changes, commits and pipeline status, as well as a link to a live instance of the app to easily preview developer's changes. Review Apps provide a live staging environment for each MR to preview changes – no more waiting on approvals or for others to provision environments for you.

The merge request is also a collaboration tool that allows team members to review each others' code and provide feedback on potential changes. Everyone has visibility into the changes and a wide variety of test results and reports are available in the merge requests. You can define approval rules to prevent merging any changes if a security report contains vulnerabilities or if a compliance report contains a denied license.



GITLAB RUNNERS

For teams using GitLab's SaaS version, GitLab offers Linux and Windows shared runners hosted on GitLab.com for executing your pipelines. This is the version that Tanuki Pets is currently using.

But what if Tanuki Pets wants to move GitLab to their on-premise data center and use their own infrastructure in the future? GitLab Runner is open source and written in Go. Once Runner software is installed, it will download a Docker image with all the dev tools in it and run the job inside the image, so minimal to no extra maintenance of the runners is required. Each job will run on a clean image for increased security and stability.

According to a Gartner report, it's important for security to be included in the DevSecOps lifecycle in small, actionable steps so that developers can react quickly. This allows the pace of security fixes to match the pace of development. The best way to bring security scanning into the development process is by using a tool like GitLab that allows developers to stay in the same platform or interface they're already using to commit, scan, and ship code to production. This makes the security process automatic every time there is a code update.

The team at Tanuki Pets needed a way to deliver new features to the market faster to stay competitive. They couldn't compromise on quality, security, or compliance, so they chose to use GitLab. Every member of the Tanuki Pets team could use GitLab's single interface to collaborate and innovate faster.

We've created the pipeline configuration, added tests to it, but how do we actually run the pipeline? Do we need to maintain dedicated build machines?

GitLab Runner can also be enabled for autoscaling so that instances can spin up or down as needed. Learn how the team at Substrakt Health was able to save 90% on their EC2 costs by autoscaling runners.

Read their story

Why GitLab

GitLab is an open DevOps platform that unifies development, operations, and security teams into a single application. With GitLab, DevSecOps architecture is built into the CI/CD process. Every merge request is scanned through its pipeline for security issues and vulnerabilities in the code and its dependencies using automated tests.

GitLab helps teams accelerate software delivery from weeks to minutes while reducing development costs and security risks.

BI Worldwide increases deployments to 10 times per day

> “One tool for SCM+CI/CD was a big initial win. Now wrapping security scans into that tool as well has already increased our visibility into security vulnerabilities. The integrated Docker registry has also been very helpful for us. Issue/Product management features let everyone operate in the same space regardless of role.”
>
> – Adam Dehnel, Product Architect, BI WORLDWIDE

Read their story

The benefits of GitLab Ultimate for DevSecOps

Dev and Ops teams working together

- Everyone working in the same system
- Smart feedback loops
- No messy integrations or plugins

Get value to customers faster

- Issues addressed earlier
- Fast cycle times
- Quality control

Reduce security and compliance risk

- Every change is fully tested and secure
- Audit logs for every action
- Single sign-on and datastore

In this eBook, we hope you've learned about some of the core capabilities that help millions of users around the world to develop better applications faster:

- Get started with GitLab, registration, creating a group and a project
- Enable Auto DevOps - GitLab automatically configure the CI/CD for you, based on years of experience and best practices from thousands of customers
- Manually configuration of CI/CD with YAML
- Add security scans and test automation
- Deploy test environment on Kubernetes
- Review test results
- GitLab runners - the agents that run the jobs.

About the Author

Itzik Gan-Baruch is a veteran of creating creative technical content which bridges gaps between business and technology, and helps non-technical users understand the value of technology. He creates software demo systems, demo tools, YouTube videos, analysts demos, plans content for demo booths at tech shows, and speaks at industry events. He is a senior technical marketing manager at GitLab with over 21 years of experience in the IT industry focusing on application delivery, agile and software testing, and is a father of three.

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 times through more efficient collaboration and enhanced focus.

Built on open source, GitLab works alongside its growing community, which is composed 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. All-remote since 2014, GitLab has more than 1,300 team members in 68 countries.

GitLab


GitLab