Intro to GitLab

DevOps on a Shell Script Budget

Goal: show how GitLab is an attactive option for DevOps workflows

Especially for smaller organizations with smaller budgets

Mac Justice

Slack, Tweets, etc.:
@macjustice

Joke about name

SYNAPSE

What's GitLab?

I'm going to break this into two parts

An overview of GitLab, the product

How I use GitLab

How I got into using GitLab, and what I do with it

What's GitLab?

hosting git repos

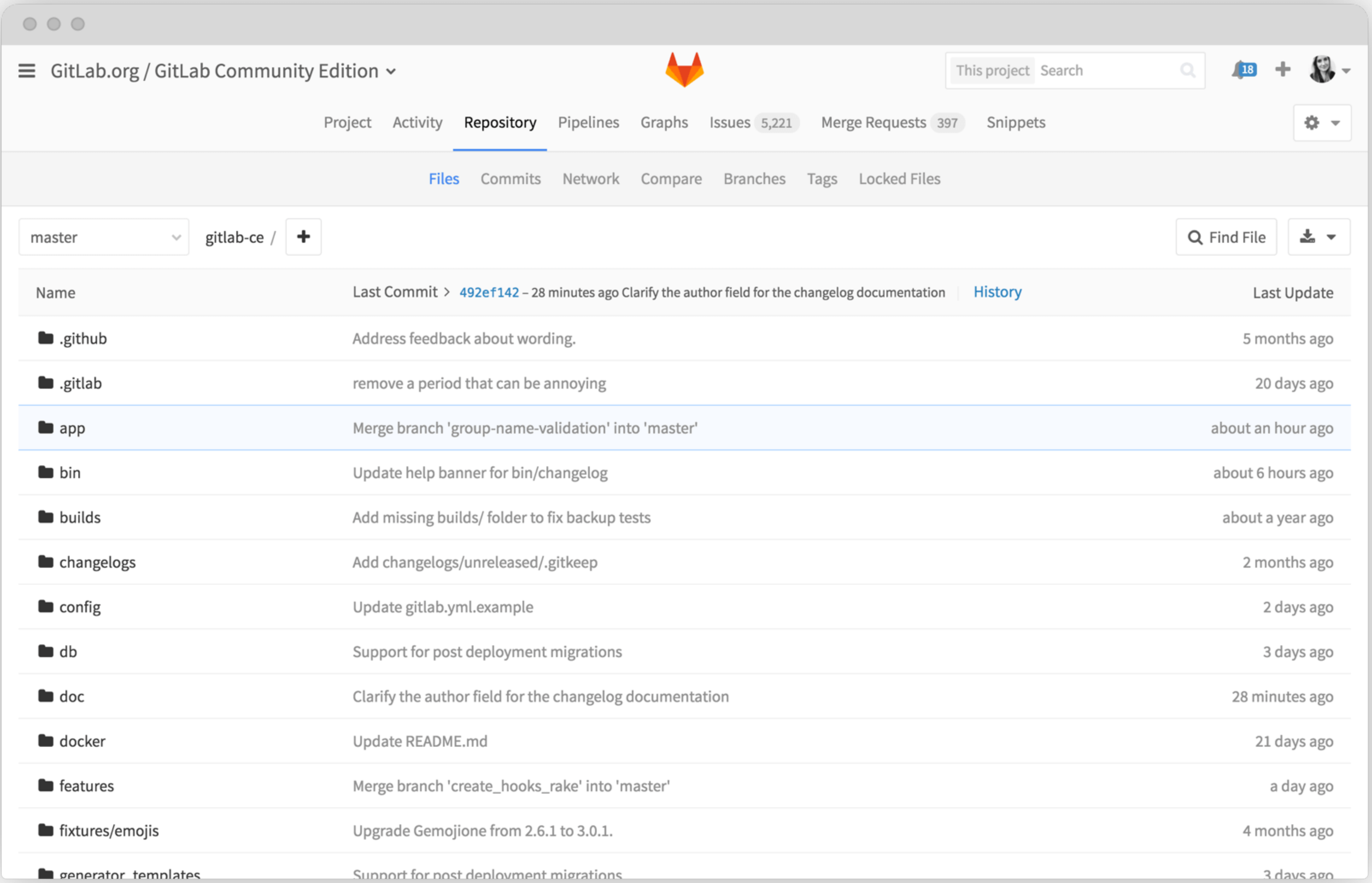
GitHub competitor

Open Source, developed in the open on GitLab.com

Gitlab.com

Self-Hosted < Most popular

Freemium



Name	Last Commit > 492ef142 – 28 minutes ago Clarify the author field for the changelog documentation History	Last Update
.github	Address feedback about wording.	5 months ago
.gitlab	remove a period that can be annoying	20 days ago
app	Merge branch 'group-name-validation' into 'master'	about an hour ago
bin	Update help banner for bin/changelog	about 6 hours ago
builds	Add missing builds/ folder to fix backup tests	about a year ago
changelogs	Add changelogs/unreleased/.gitkeep	2 months ago
config	Update gitlab.yml.example	2 days ago
db	Support for post deployment migrations	3 days ago
doc	Clarify the author field for the changelog documentation	28 minutes ago
docker	Update README.md	21 days ago
features	Merge branch 'create_hooks_rake' into 'master'	a day ago
fixtures/emojis	Upgrade Gemojione from 2.6.1 to 3.0.1.	4 months ago
generator_templates	Support for post deployment migrations	3 days ago

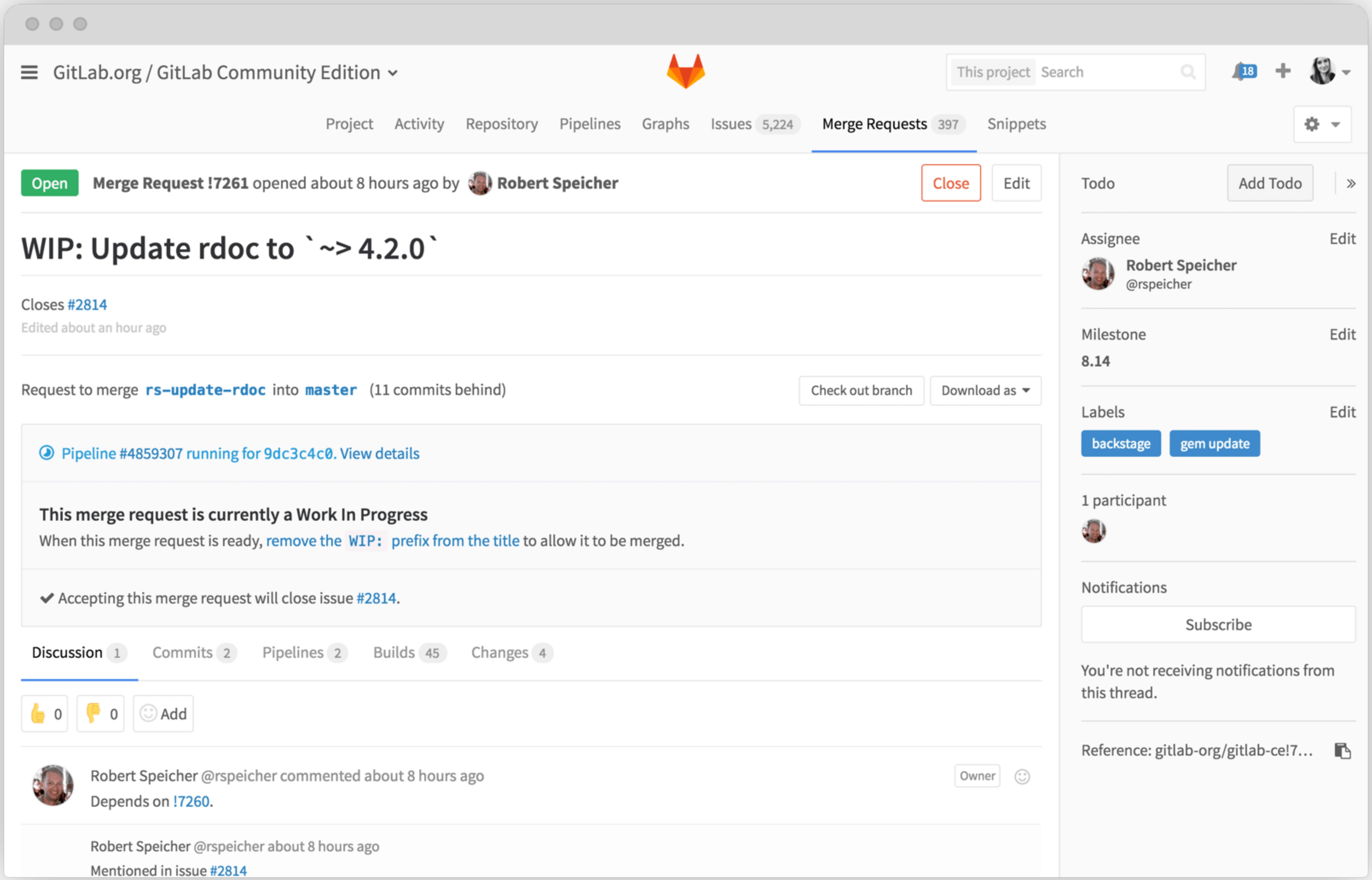
Here's some of the high level features

Just like with GitHub, you can store git repos there

Clone, push, etc.

Browse and edit in browser

Definition of "Project" in GitLab



Open **Merge Request !7261** opened about 8 hours ago by **Robert Speicher**

WIP: Update rdoc to `~> 4.2.0`

Closes #2814

Edited about an hour ago

Request to merge `rs-update-rdoc` into `master` (11 commits behind)

Pipeline #4859307 running for `9dc3c4c0`. View details

This merge request is currently a Work In Progress

When this merge request is ready, remove the `WIP:` prefix from the title to allow it to be merged.

✓ Accepting this merge request will close issue #2814.

Robert Speicher @rspeicher commented about 8 hours ago

Depends on !7260.

Robert Speicher @rspeicher about 8 hours ago

Mentioned in issue #2814

Assignee

Robert Speicher

@rspeicher

Milestone

8.14

Labels

backstage gem update

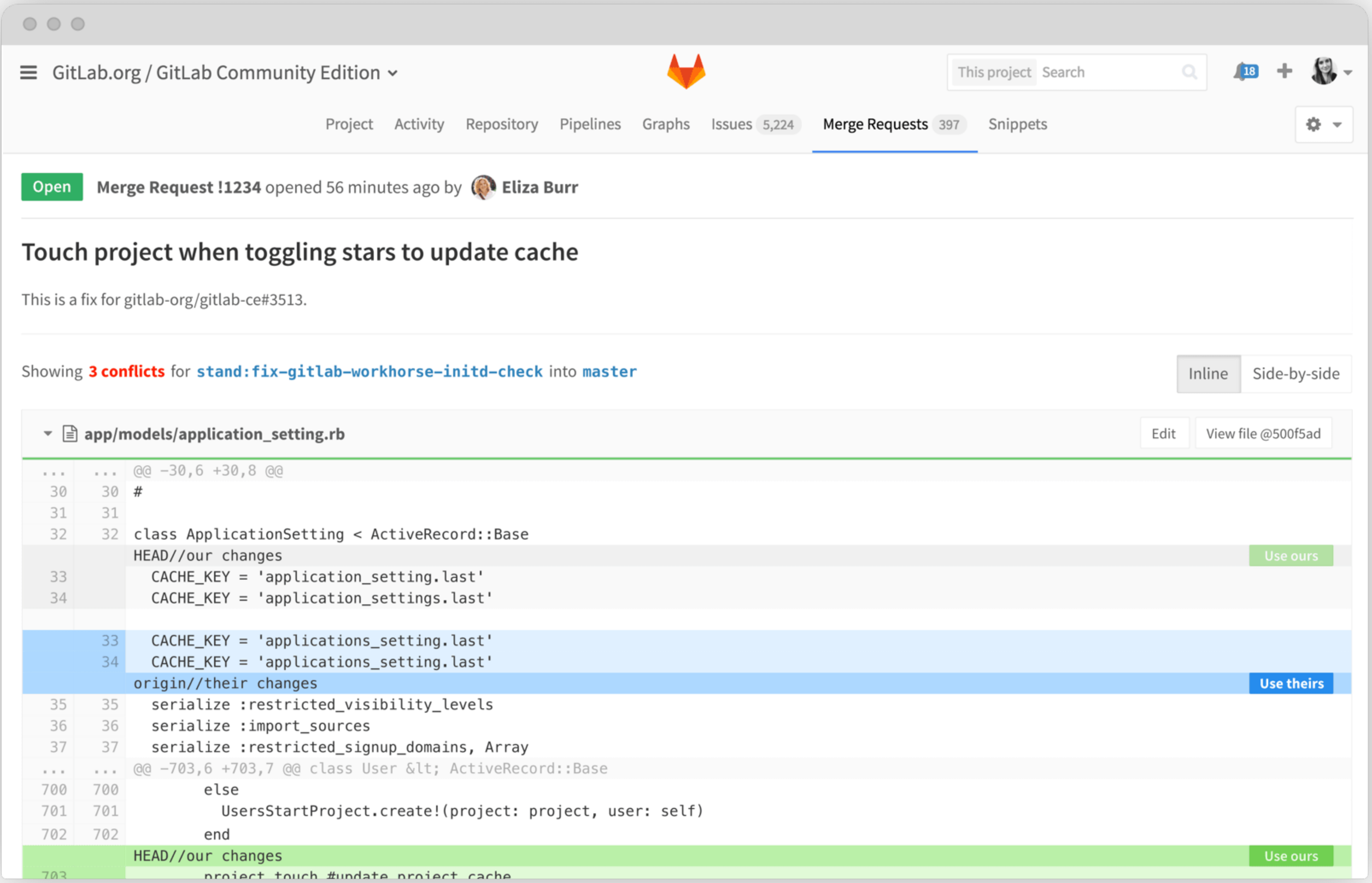
1 participant

Notifications

You're not receiving notifications from this thread.

Reference: gitlab-org/gitlab-ce!7...

Again like GitHub, you can create merge requests, so you and your team can review changes going into production.



Merge Request !1234 opened 56 minutes ago by Eliza Burr

Touch project when toggling stars to update cache

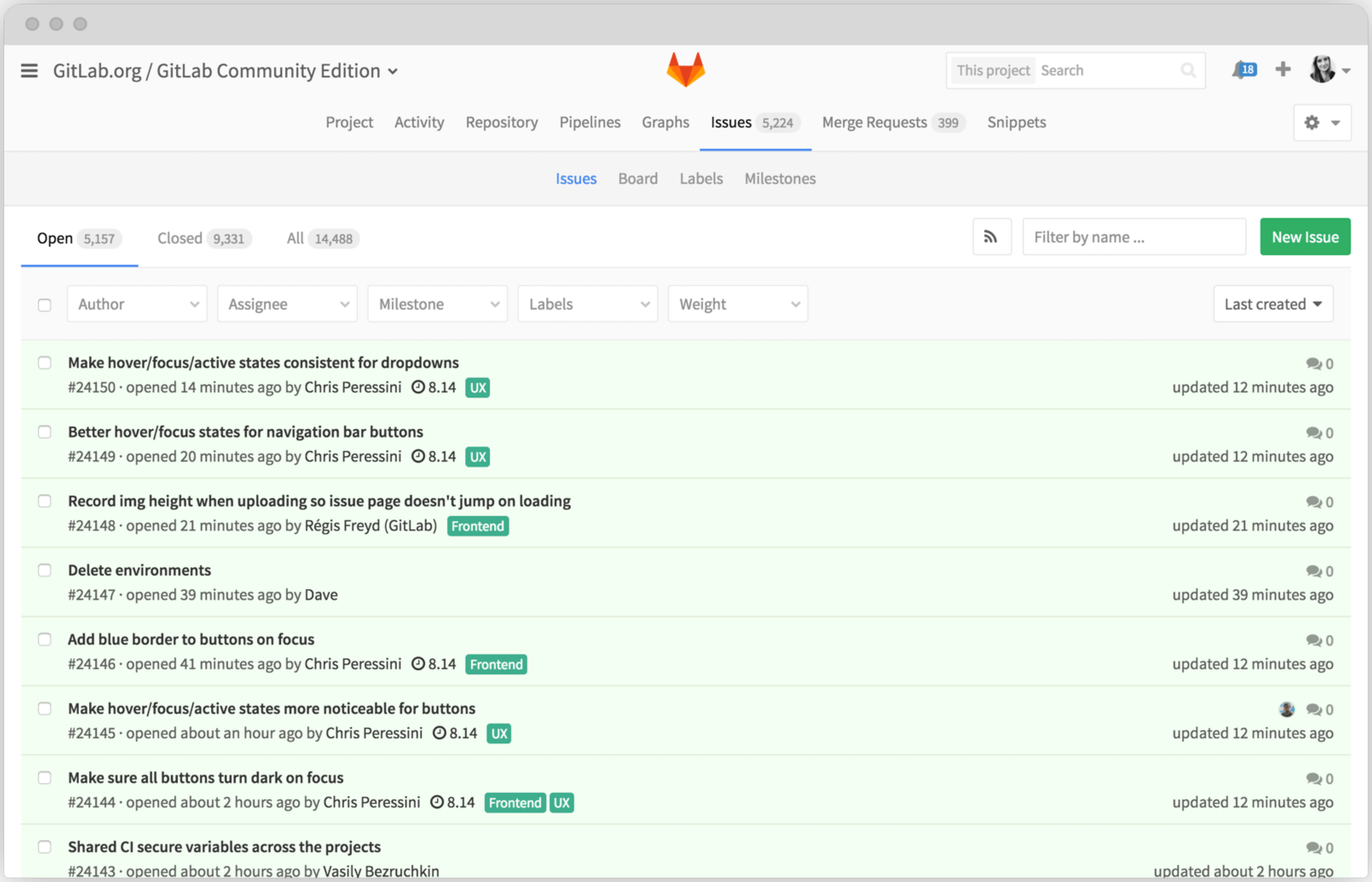
This is a fix for gitlab-org/gitlab-ce#3513.

Showing 3 conflicts for stand:fix-gitlab-workhorse-initd-check into master

app/models/application_setting.rb

```
@@ -30,6 +30,8 @@
#

class ApplicationSetting < ActiveRecord::Base
HEAD//our changes
  CACHE_KEY = 'application_setting.last'
  CACHE_KEY = 'application_settings.last'

  CACHE_KEY = 'applications_setting.last'
  CACHE_KEY = 'applications_setting.last'
origin//their changes
  serialize :restricted_visibility_levels
  serialize :import_sources
  serialize :restricted_signup_domains, Array
@@ -703,6 +703,7 @@ class User &lt; ActiveRecord::Base
        else
          UsersStartProject.create!(project: project, user: self)
        end
HEAD//our changes
        project.touch #update project cache
```

If your merges have conflicts, you can resolve them right in the browser



Make hover/focus/active states consistent for dropdowns
#24150 · opened 14 minutes ago by Chris Peressini 8.14 UX — 0 — updated 12 minutes ago

Better hover/focus states for navigation bar buttons
#24149 · opened 20 minutes ago by Chris Peressini 8.14 UX — 0 — updated 12 minutes ago

Record img height when uploading so issue page doesn't jump on loading
#24148 · opened 21 minutes ago by Régis Freyd (GitLab) Frontend — 0 — updated 21 minutes ago

Delete environments
#24147 · opened 39 minutes ago by Dave — 0 — updated 39 minutes ago

Add blue border to buttons on focus
#24146 · opened 41 minutes ago by Chris Peressini 8.14 Frontend — 0 — updated 12 minutes ago

Make hover/focus/active states more noticeable for buttons
#24145 · opened about an hour ago by Chris Peressini 8.14 UX — 0 — updated 12 minutes ago

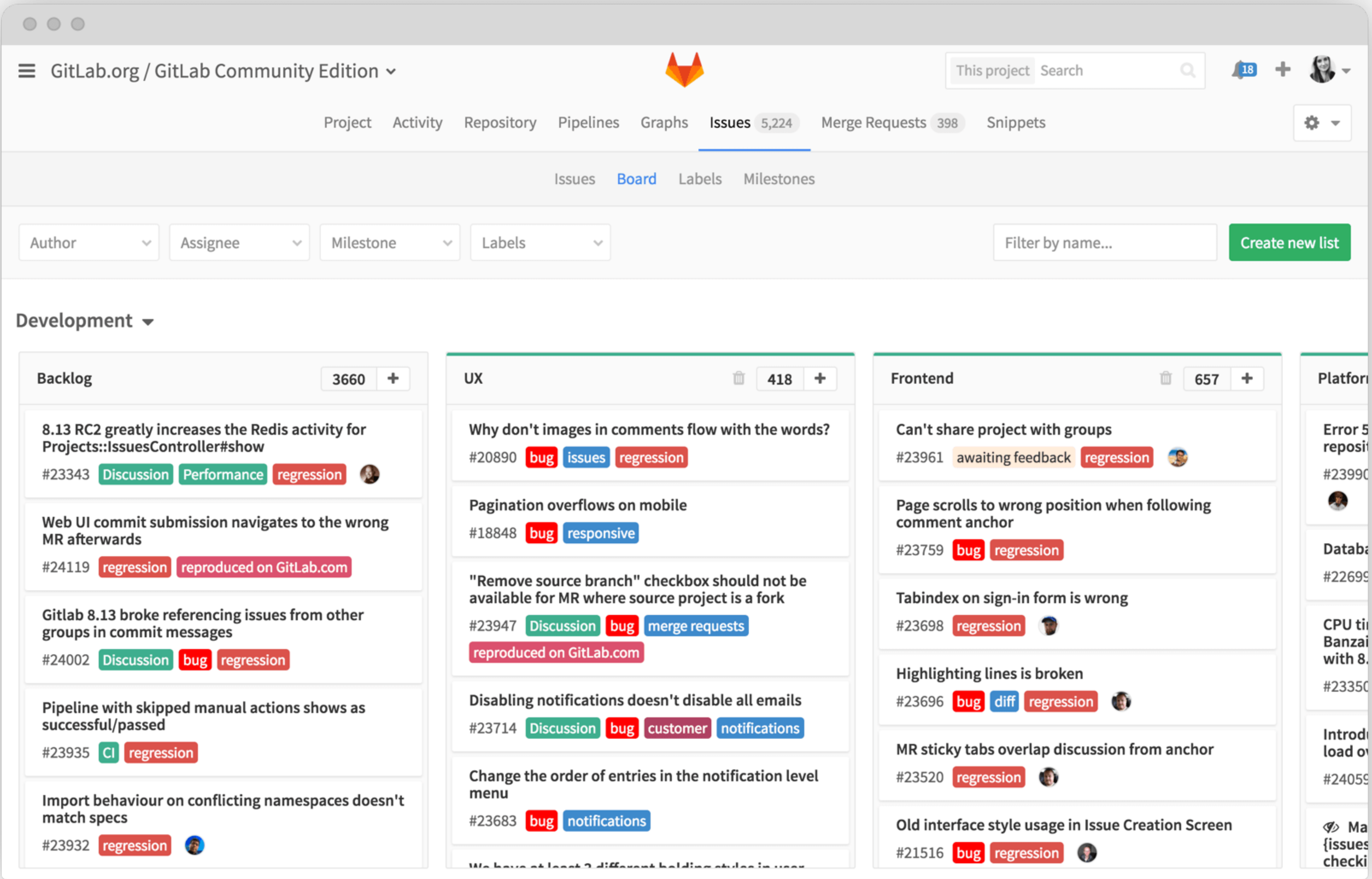
Make sure all buttons turn dark on focus
#24144 · opened about 2 hours ago by Chris Peressini 8.14 Frontend UX — 0 — updated 12 minutes ago

Shared CI secure variables across the projects
#24143 · opened about 2 hours ago by Vasily Bezruchkin — 0 — updated about 2 hours ago

Built in Issue Tracker, like Jira or FogBugz

You can attach issues to commits, CI builds, merge requests

Assign them to members of your team



Backlog 3660

8.13 RC2 greatly increases the Redis activity for Projects::IssuesController#show
#23343 Discussion Performance regression

Web UI commit submission navigates to the wrong MR afterwards
#24119 regression reproduced on GitLab.com

Gitlab 8.13 broke referencing issues from other groups in commit messages
#24002 Discussion bug regression

Pipeline with skipped manual actions shows as successful/passed
#23935 CI regression

Import behaviour on conflicting namespaces doesn't match specs
#23932 regression

UX 418

Why don't images in comments flow with the words?
#20890 bug issues regression

Pagination overflows on mobile
#18848 bug responsive

"Remove source branch" checkbox should not be available for MR where source project is a fork
#23947 Discussion bug merge requests reproduced on GitLab.com

Disabling notifications doesn't disable all emails
#23714 Discussion bug customer notifications

Change the order of entries in the notification level menu
#23683 bug notifications

Frontend 657

Can't share project with groups
#23961 awaiting feedback regression

Page scrolls to wrong position when following comment anchor
#23759 bug regression

Tabindex on sign-in form is wrong
#23698 regression

Highlighting lines is broken
#23696 bug diff regression

MR sticky tabs overlap discussion from anchor
#23520 regression

Old interface style usage in Issue Creation Screen
#21516 bug regression

Platfor

Error 5
reposi
#23990

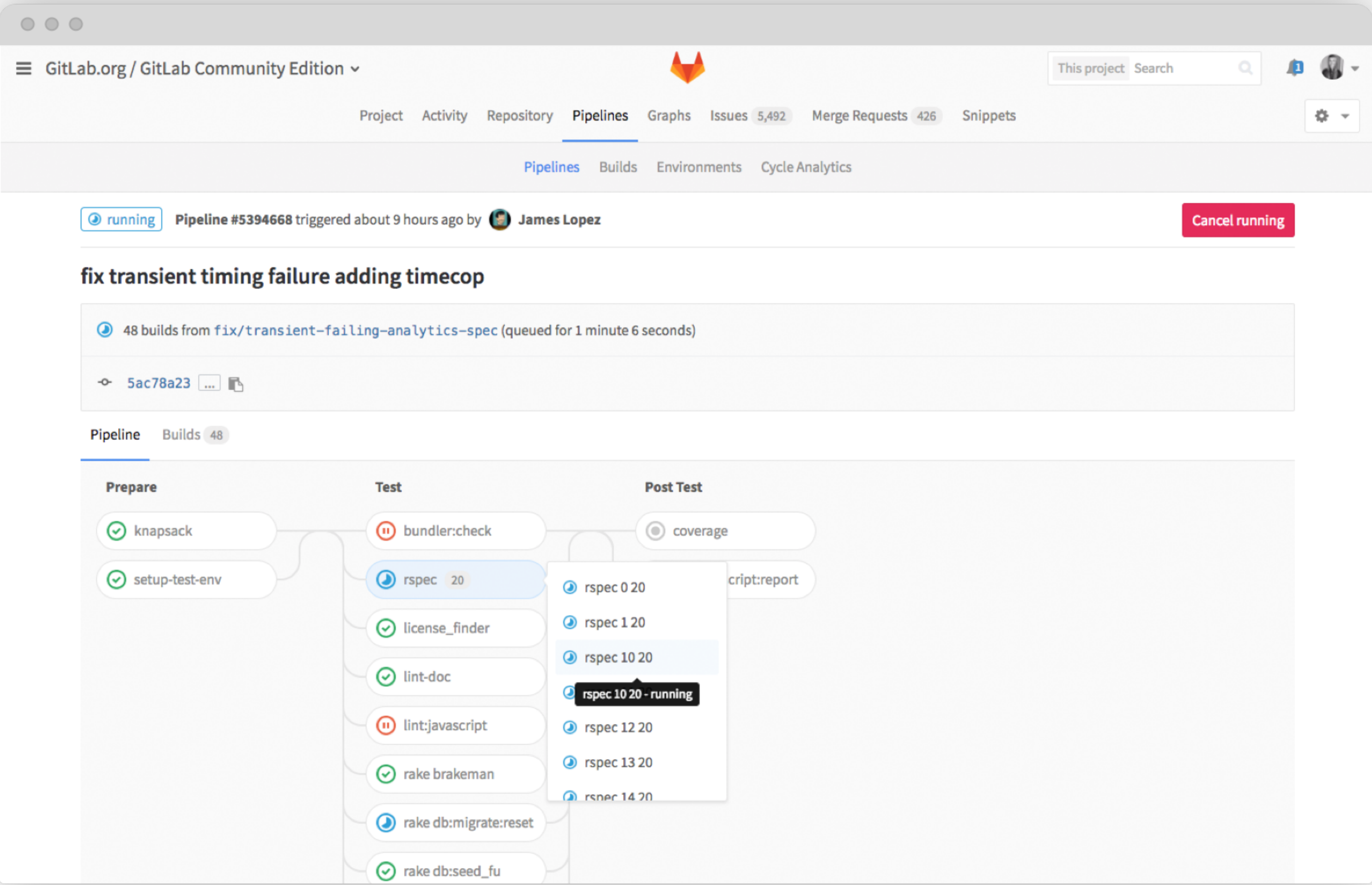
Databa
#22699

CPU ti
Banzai
with 8.
#23350

Introd
load o
#24059

Ma
{issues
checki

And there's even a neat Trello style Kanban board for visualizing your open issues.



Pipeline #5394668 triggered about 9 hours ago by James Lopez

fix transient timing failure adding timecop

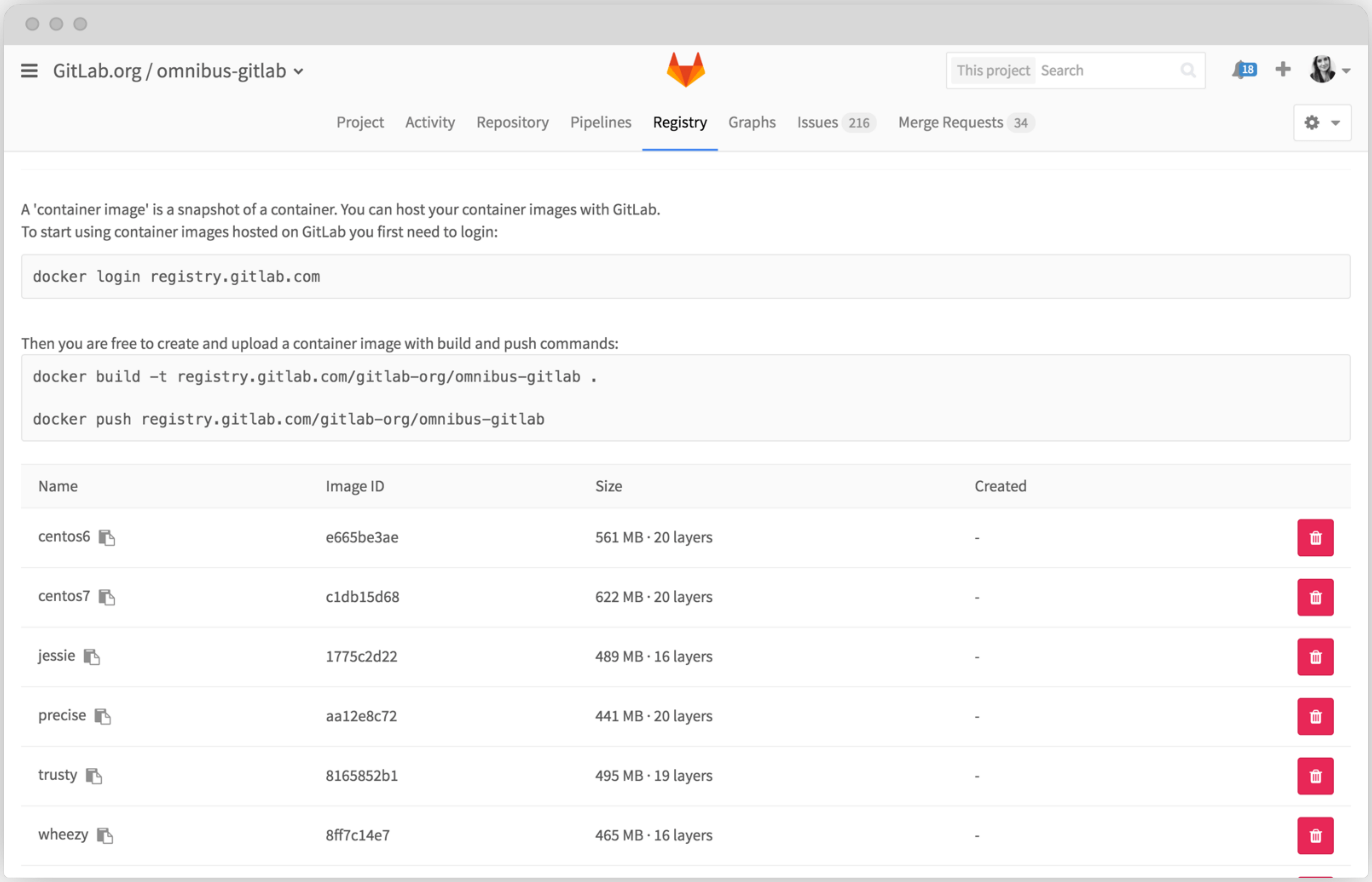
48 builds from fix/transient-failing-analytics-spec (queued for 1 minute 6 seconds)

5ac78a23

One of my personal favorites is the CI pipeline tool

Like Jenkins, but different

more to come



A 'container image' is a snapshot of a container. You can host your container images with GitLab.
To start using container images hosted on GitLab you first need to login:

```
docker login registry.gitlab.com
```

Then you are free to create and upload a container image with build and push commands:

```
docker build -t registry.gitlab.com/gitlab-org/omnibus-gitlab .

docker push registry.gitlab.com/gitlab-org/omnibus-gitlab
```

Name	Image ID	Size	Created
centos6	e665be3ae	561 MB · 20 layers	-
centos7	c1db15d68	622 MB · 20 layers	-
jessie	1775c2d22	489 MB · 16 layers	-
precise	aa12e8c72	441 MB · 20 layers	-
trusty	8165852b1	495 MB · 19 layers	-
wheezy	8ff7c14e7	465 MB · 16 layers	-

And for kicks there's even a container registry

How do I get it?

So let's talk really quick about what it takes to run your own GitLab server.

Installation

- "Omnibus" package for Linux
- Docker Container
- Pre-built VMs (Amazon EC2/LightSail, Digital Ocean)

At Synapse we host GitLab on an Ubuntu VM in our VMware cluster, most common Linux flavors are supported

Docker container

Reccomend trying AWS or Digital Ocean if you want a running GitLab instance set up in just a minute or two.

Raspberry Pi

Self-Hosted Pricing

Edition	Price	Support	Features
Community	Free	'Community'	All major functionality
Enterprise Starter	$40/user/year	Next Day	Finer permissions
Enterprise Premium	$200/user/year	4 Hour	High Availability, other advanced features

Community edition has all the features I'm going to describe today.

We upgraded to Starter after 5 years of use for support and some additional permissions control

Authentication & Authorization

- LDAP
- OAuth & SAML
- Kerberos

More than one

At Synapse, we use G Suite SAML for employee login, cross-referenced with LDAP for group permissions.

Synapse customers use Google Oauth via the OmniAuth feature.

Integrations

Chat

Prebuilt integrations for the major chat services

So you can get notifications for pipelines completed or issues assigned to you

Integrations

External CI

You're not required to use GitLab's CI, if you already have one you like you can tie it in.

Integrations

Kubernetes

I don't use Kubernetes myself, but GL has been spending a lot of resources making sure GL is closely integrated.

How I use GitLab

That's the general overview, now I'm going to give you details on how I use it.

The firmware developers I work with need a version control system, and since many work with embedded Linux they're very comfortable with git.

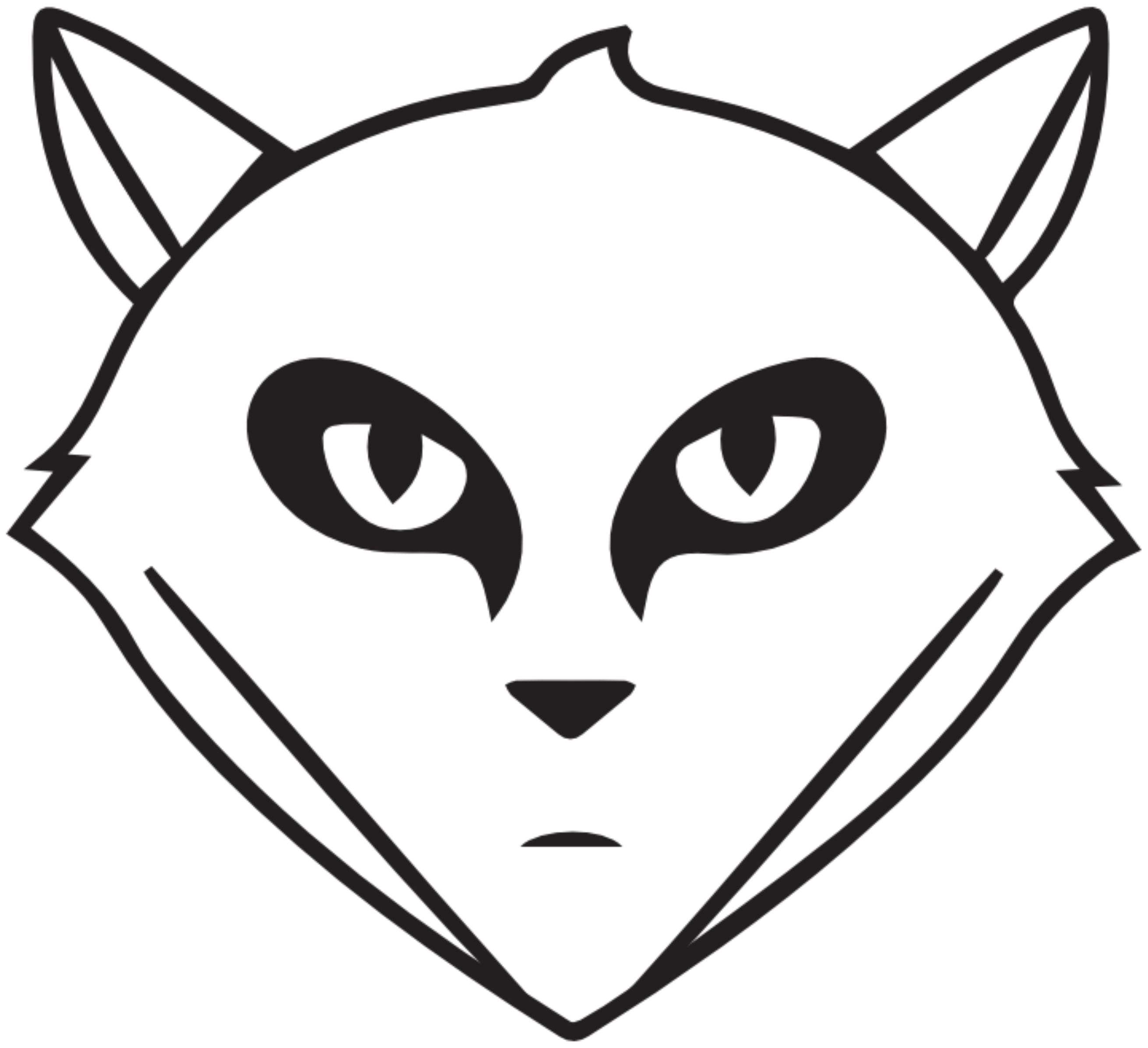
So, they looked into setting up a central git repo host.

2009?

First, Synapse used Gitolite, a free open source tool which gets the job done but with very few frills.

It has no GUI, so user interaction is strictly via ssh. It's pretty much just a git host with access control features.

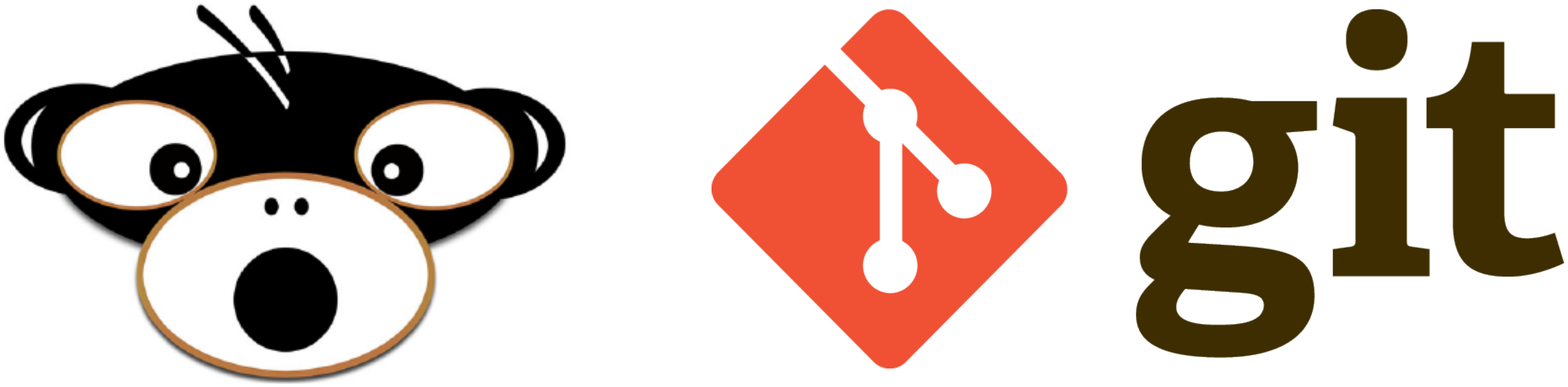
2012



Around 2012, Synapse was getting bigger, we needed a better tool: collaboration, web UI, etc.

We picked GitLab largely because it was free and easy to set up.

Also because the guy doing the choosing liked Rails.



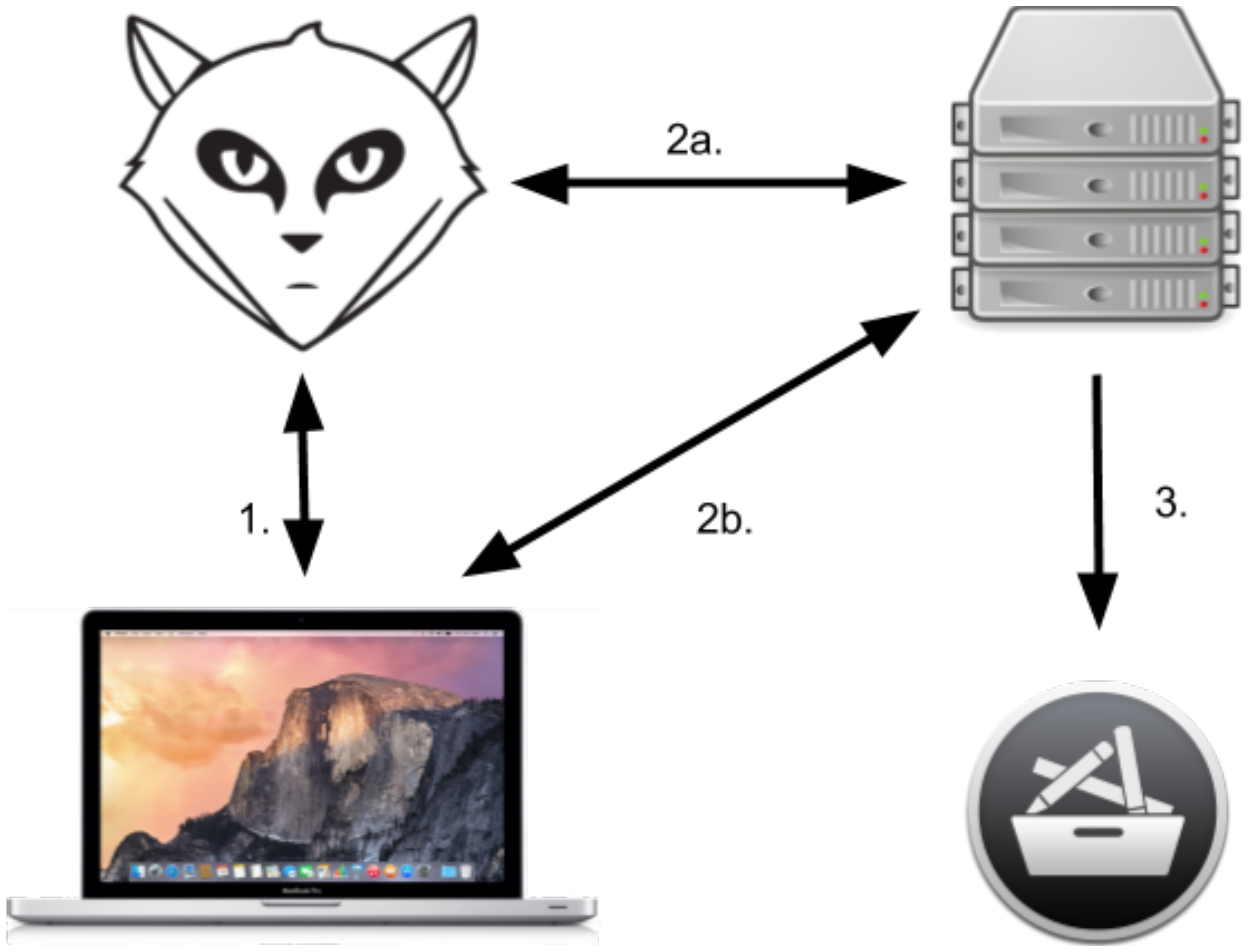
In the following years, I had set up Munki and was interested in keeping my repo in git.

If you're not familiar with how Munki is structured, a repo is generally composed of a bunch of plist files and packages.

The plists are easy to track with git

But git doesn't like big files like packages.

2014: GitLab & Munki 1.0



1. Push plists to GitLab
2. SSH to Munki Server
    a. Pull from GitLab
    b. Rsync pkgs from admin
3. Clients get updates!?

Here's my first git-enabled munki setup

Problems

Very manual

Easy to accidentally conflict pkg changes with another user, if they didn't sync up with the git host AND server first

INTERLUDE: Git LFS

Other people wanted to track big files with git, too

Git Fat, Git Annex

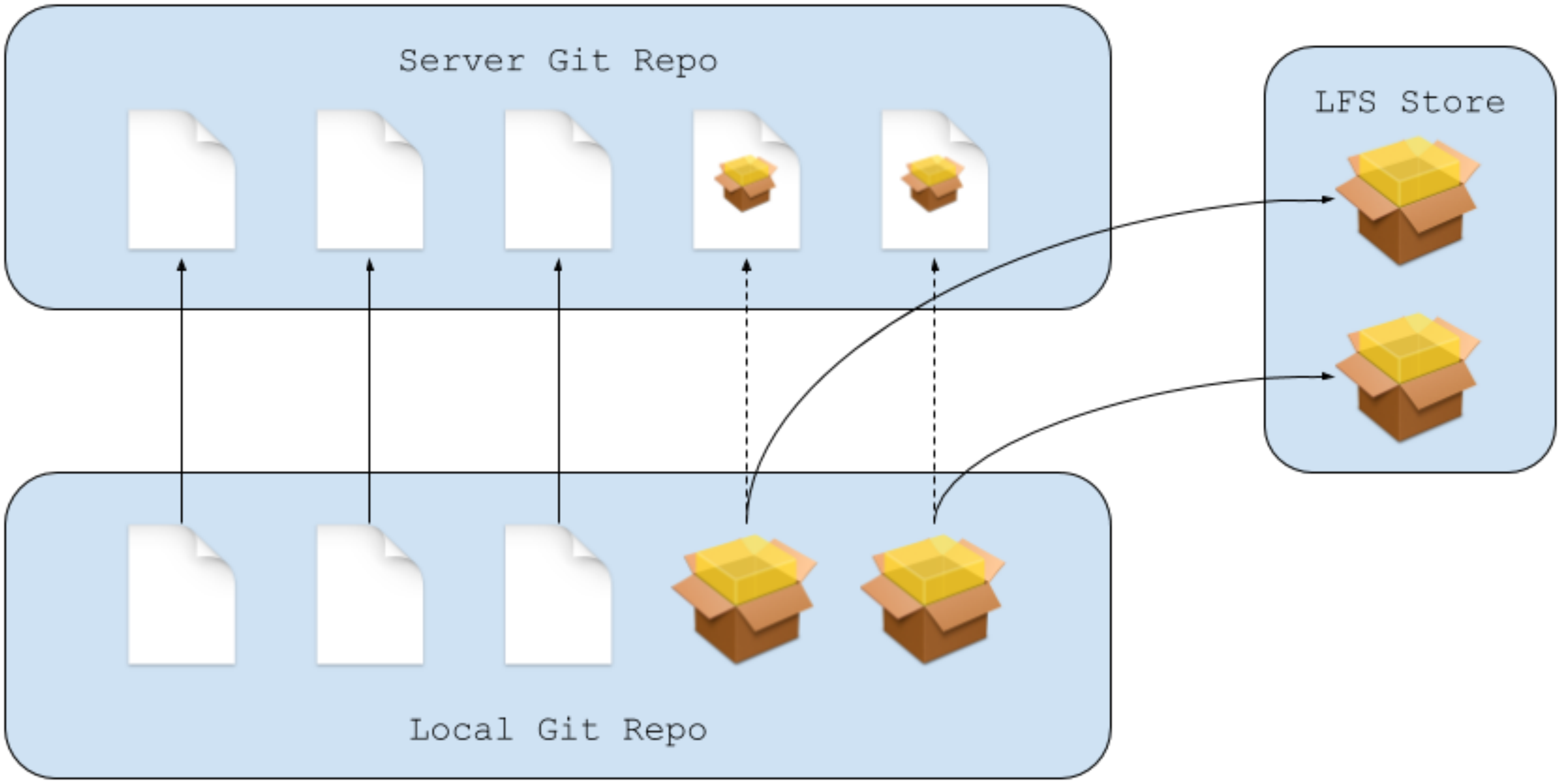
GitHub announced Git LFS, and shortly afterward GitLab announced they would support it.

Git + 📦💽 = 😰

Because Git is designed for text files, it's not great at managing binary files, especially large ones.

`Git LFS + 📦💽 = 😍`

Git LFS, however, does just fine



Here's a rough sketch of how LFS works.

Enabling it in a GitLab project is as simple as clicking a checkbox and making sure you have sufficient space on your server. You can limit how much space each project has for LFS storage.

Git LFS can handle pretty big files. The biggest I've had cause to use was a 7.7GB El Capitan AutoDMG image, which was no trouble at all.

Git LFS Example

Local Installation

```
brew install git-lfs
git lfs install
```

Installing git lfs is easy. It's available on brew and MacPorts, or you can download the binary from GitHub.

the second command modifies your global git config to support LFS

Git LFS Example

Repo Setup

```
cd munki-repo
git lfs track "*.pkg"
git add .gitattributes
git commit -m "Added LFS tracking for PKGs"
```

Now that LFS is installed, you can just go to a repo and tell it which files to track. The pattern is added to .gitattributes, and once you commit that change you're in business.

Git LFS Example

Everyday Use

```
git add pkgs/SweetApp.pkg
git commit -m "Added SweetApp"
git push origin master
```

From here it's as easy as using standard git commands to commit, push and pull.

I tend to use MunkiAdmin for day to day work, so I make my changes there and commit them in a separate git app. Fork is my current favorite.

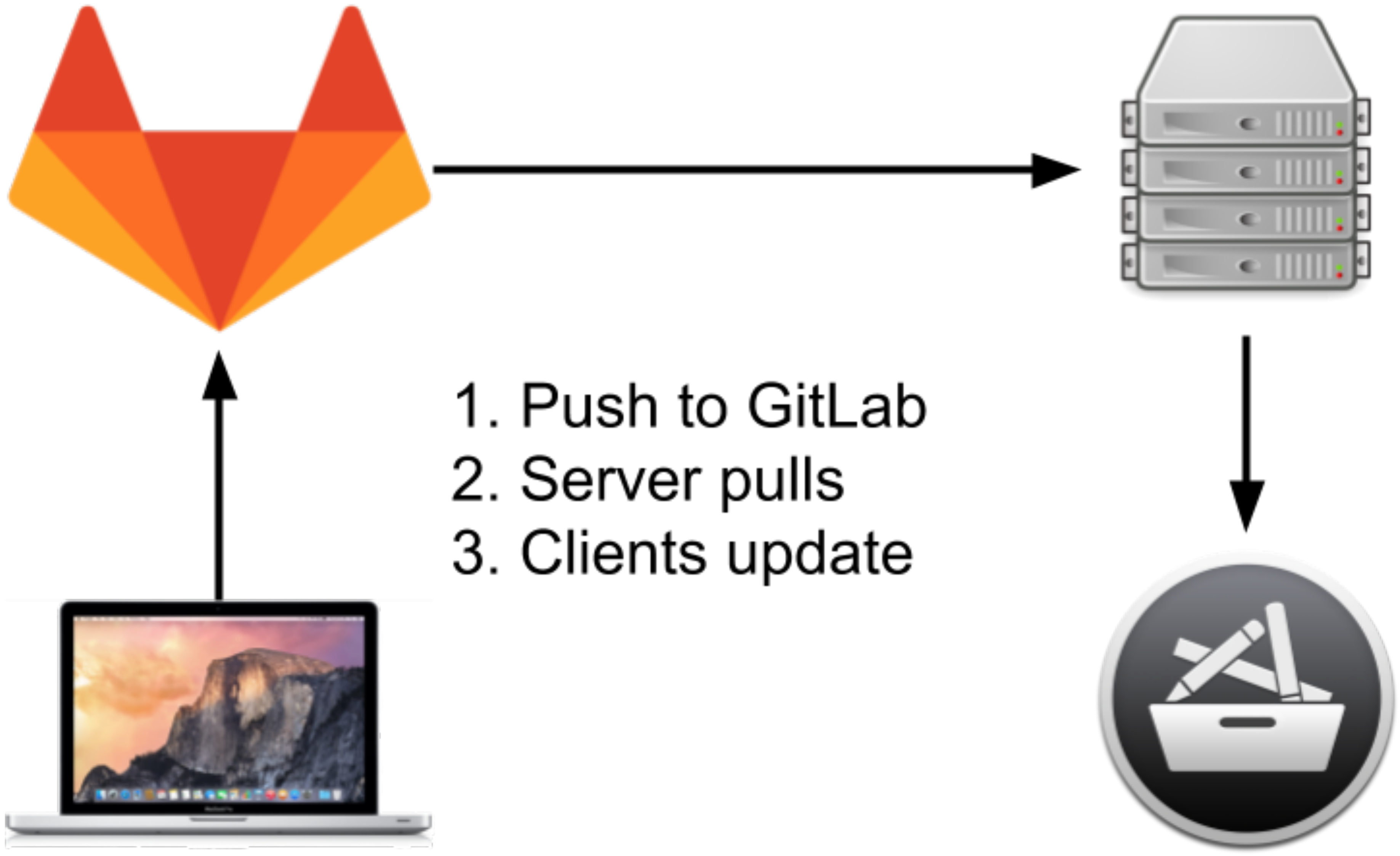
2016

Back to the story

Our GitLab hadn't gotten any attention for a few years, so I got it upgraded to current, with the new logo, so I could get some of that sweet sweet LFS

First thing I did was rebuild my Munki workflow.

2016



1. Push to GitLab
2. Server pulls
3. Clients update

Here was the new workflow. It was pretty good!

Cron on server

Everything was in LFS

Collaboration worked!

INTERLUDE: CI

Now another interlude, I want to explain CI, and GitLab's approach.

Continuous Integration

1. Push code to GitLab ⌨️ ⬆️
2. Do something with that code ⚙️ 🛠️
3. Report results 🚫 ✅

In software development, CI means you push your code, and it gets tested.

Essentially though, CI tools are really a simple automation tool

When X happens, do Y

Continuous Deployment/ Delivery

1. Push code to GitLab ⌨️ ⬆️

2. Test that code ⚗️ 🔬 🚫 ✅

3. If the tests pass, put it into production ⚙️ 🛠️

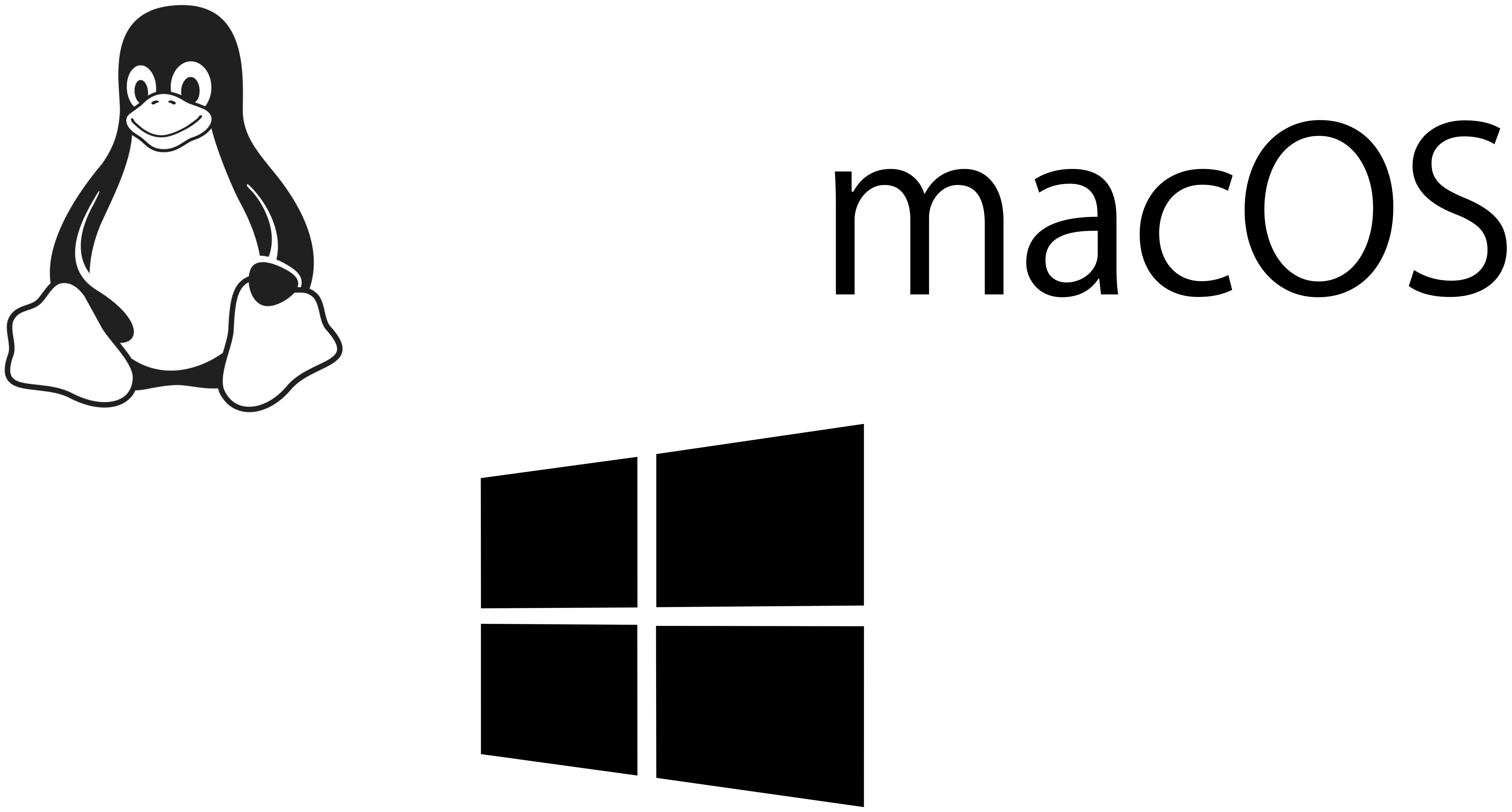
The difference between CI and CD is more conceptual than anything.

The main difference between these is whether you want a human at the end giving the thumbs up.

CI Runners

The tools that actually execute the CI tasks are called runners in GitLab.

CI Runners



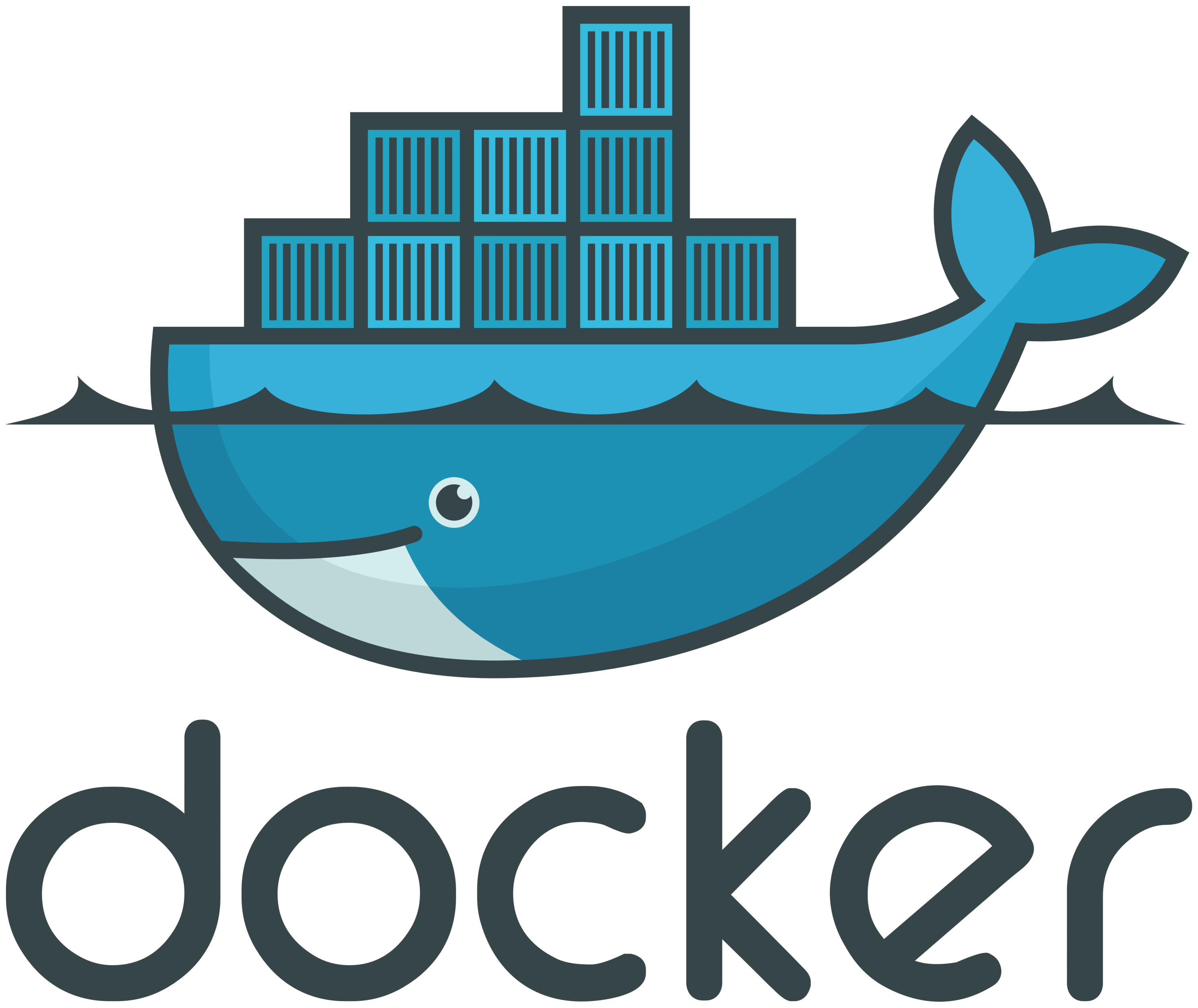
There are runners available for every major OS.

Once you have the runner installed, it will wait for GitLab to assign it jobs. Jobs can be run in sequence or parallel, and can be anything you can script.

Runners can be shared by many GitLab projects, or you can create runners reserved for specific projects, if a specific environment is needed.

You can tag runners, so if a certain job needs a certain environment, you tag the runner that is configured for it

CI Runners



GitLab CI even supports Docker.

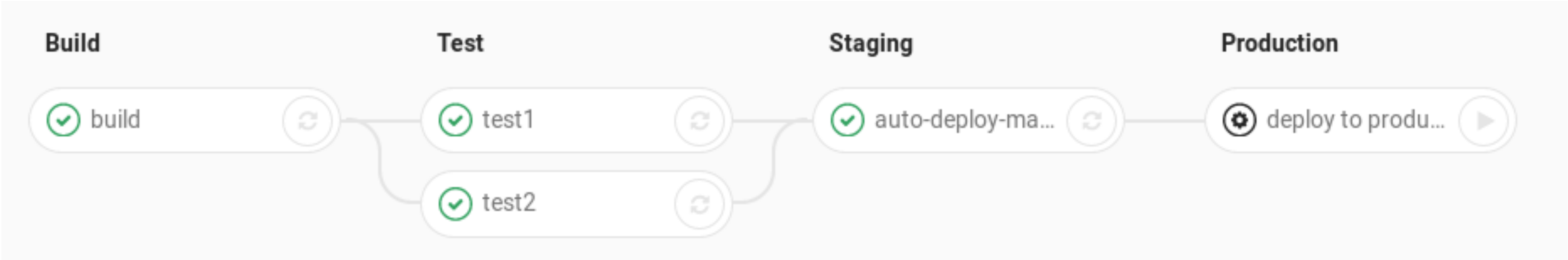
When you configure a runner to use docker, it will start a container of your choosing to excute a job in.

This is really handy, there's prebuilt containers with Python or AWS tools ready to use, and you can build your own and store them on GitLab for use.

CI Runner Autoscaling

The GitLab runner also has an "autoscale" mode, which uses Docker Machine to create temporary VMs in a cloud provider or other hypervisor. The VMs execute their assigned jobs, return the results, and are terminated and deleted.

Pipelines



Each task you include in CI is called a job.

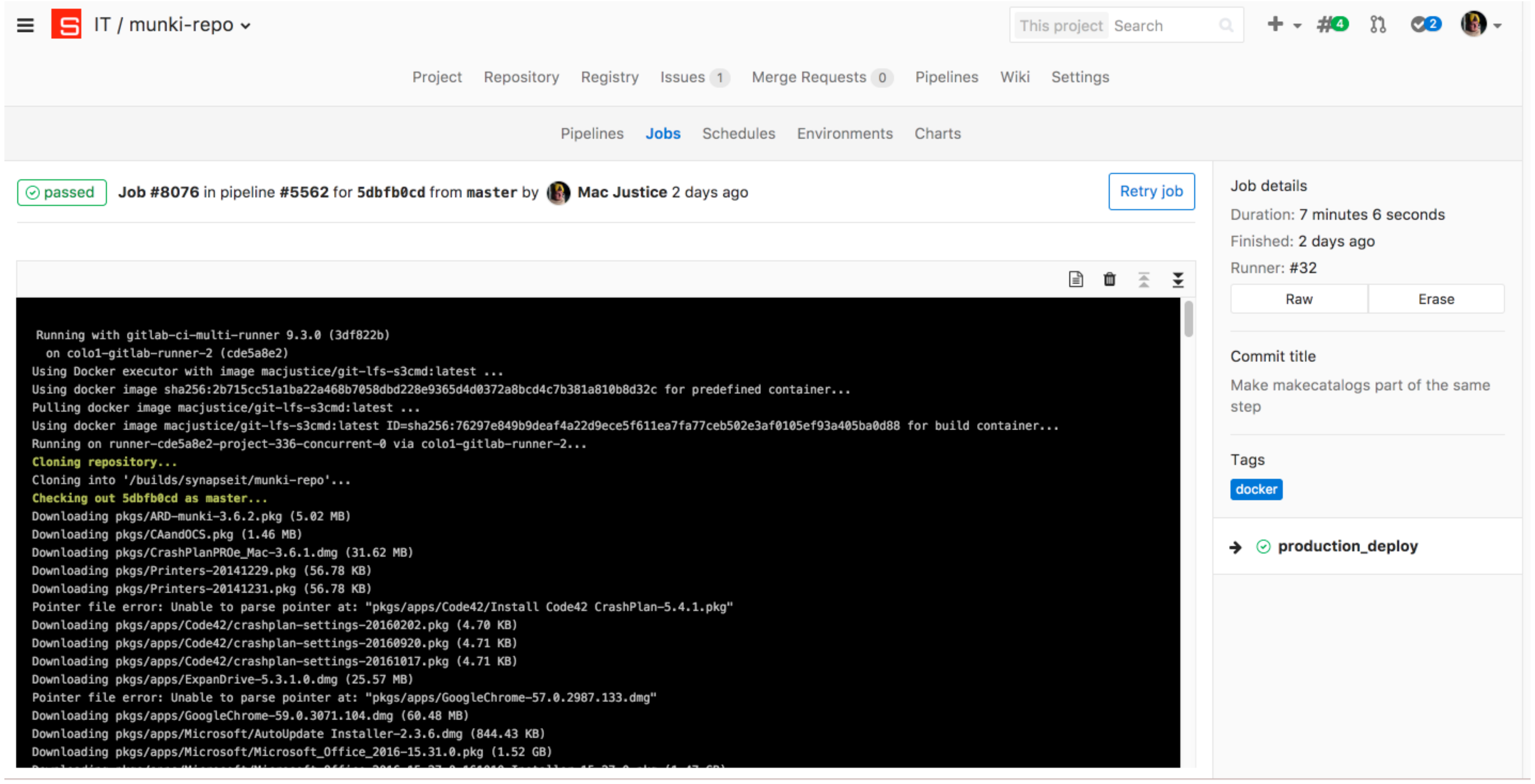
More than one job makes a pipeline.

Pipelines execute in an order you define

Sequential / Parallel

If an earlier stage in the pipeline fails, the next stage doesn't start

CI Output



passed **Job #8076** in pipeline **#5562** for **5dbfb0cd** from **master** by **Mac Justice** 2 days ago

```
 Running with gitlab-ci-multi-runner 9.3.0 (3df822b)
  on colo1-gitlab-runner-2 (cde5a8e2)
Using Docker executor with image macjustice/git-lfs-s3cmd:latest ...
Using docker image sha256:2b715cc51a1ba22a468b7058dbd228e9365d4d0372a8bcd4c7b381a810b8d32c for predefined container...
Pulling docker image macjustice/git-lfs-s3cmd:latest ...
Using docker image macjustice/git-lfs-s3cmd:latest ID=sha256:76297e849b9deaf4a22d9ece5f611ea7fa77ceb502e3af0105ef93a405ba0d88 for build container...
Running on runner-cde5a8e2-project-336-concurrent-0 via colo1-gitlab-runner-2...
Cloning repository...
Cloning into '/builds/synapseit/munki-repo'...
Checking out 5dbfb0cd as master...
Downloading pkgs/ARD-munki-3.6.2.pkg (5.02 MB)
Downloading pkgs/CAandOCS.pkg (1.46 MB)
Downloading pkgs/CrashPlanPROe_Mac-3.6.1.dmg (31.62 MB)
Downloading pkgs/Printers-20141229.pkg (56.78 KB)
Downloading pkgs/Printers-20141231.pkg (56.78 KB)
Pointer file error: Unable to parse pointer at: "pkgs/apps/Code42/Install Code42 CrashPlan-5.4.1.pkg"
Downloading pkgs/apps/Code42/crashplan-settings-20160202.pkg (4.70 KB)
Downloading pkgs/apps/Code42/crashplan-settings-20160920.pkg (4.71 KB)
Downloading pkgs/apps/Code42/crashplan-settings-20161017.pkg (4.71 KB)
Downloading pkgs/apps/ExpanDrive-5.3.1.0.dmg (25.57 MB)
Pointer file error: Unable to parse pointer at: "pkgs/apps/GoogleChrome-57.0.2987.133.dmg"
Downloading pkgs/apps/GoogleChrome-59.0.3071.104.dmg (60.48 MB)
Downloading pkgs/apps/Microsoft/AutoUpdate Installer-2.3.6.dmg (844.43 KB)
Downloading pkgs/apps/Microsoft/Microsoft_Office_2016-15.31.0.pkg (1.52 GB)
```

Job details

Duration: 7 minutes 6 seconds

Finished: 2 days ago

Runner: #32

Commit title

Make makecatalogs part of the same step

Tags

docker

production_deploy

How do you how things went?

All output from a CI job is visible in a console. Very helpful for troubleshooting.

CI Configuration

.gitlab-ci.yml

```yaml
validate:
  stage: test
  script: lint_roller.sh

roll_out:
  stage: deploy
  only: master
  script:
    - ./sound_klaxons_and_flash_lights.py
    - rsync build/* user@remote_server:/deploy/path/
```

My favorite part about GitLab CI is that you define your CI jobs as code, so changes to your build process are tracked in git too.

Here's an example of a CI config file. You just add one of these to your repo, and GitLab will try to start executing CI for your project on an available runner. You can call scripts and commands, specify stage order, set variables, pass files between stages, and lots more.

.gitlab-ci.yml, annotated

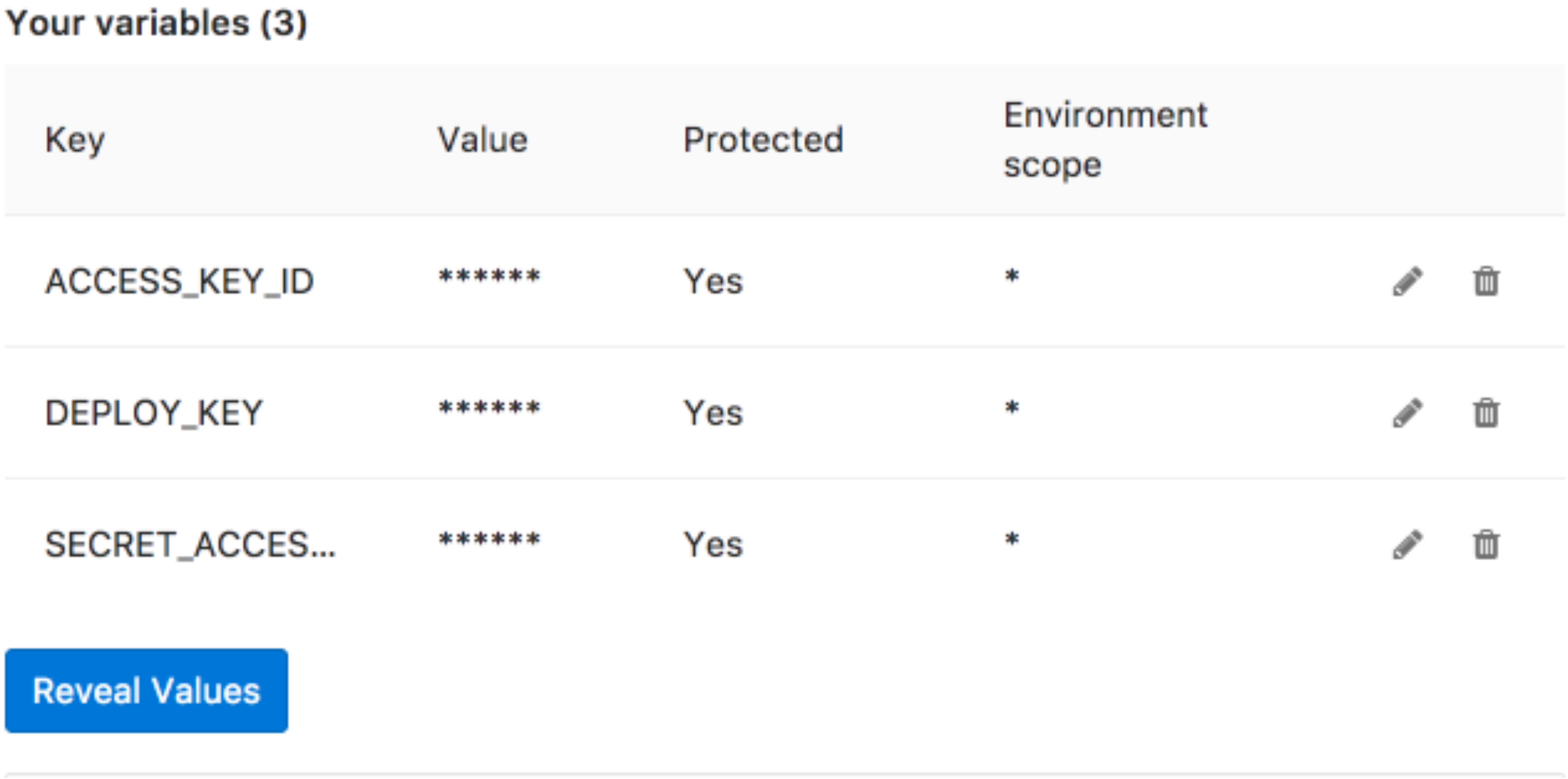
```
validate: # first job name
  stage: test # All 'test' stage jobs run before 'deploy' stage
  script: lint_roller.sh # Run this script

deploy: # second job name
  stage: deploy # Start only when all 'test' stage jobs complete
  only: master # Only run on Master branch
  script:
    - ./sound_klaxons_and_flash_lights.py # Call script in repo
    - rsync build/* user@remote_server:/deploy/path/ # inline command
```

In this example, I have two jobs, validate and roll-out. Validate is in the test stage, so it goes first, and it just runs the "check for typos" script. It will run any time someone pushes a commit to the parent GitLab project.

The second job, Roll-out, is marked as a deploy stage, so it only starts when all test jobs complete successfully. I specify to only run this job when there are updates to the master branch, because I don't want to push development branches to production. Finally, it runs a script, and an inline command.

If both jobs succeed, GitLab CI reports success.

Secret Variables



Your variables (3)

Key	Value	Protected	Environment scope		
ACCESS_KEY_ID	******	Yes	*	✎	🗑
DEPLOY_KEY	******	Yes	*	✎	🗑
SECRET_ACCES...	******	Yes	*	✎	🗑

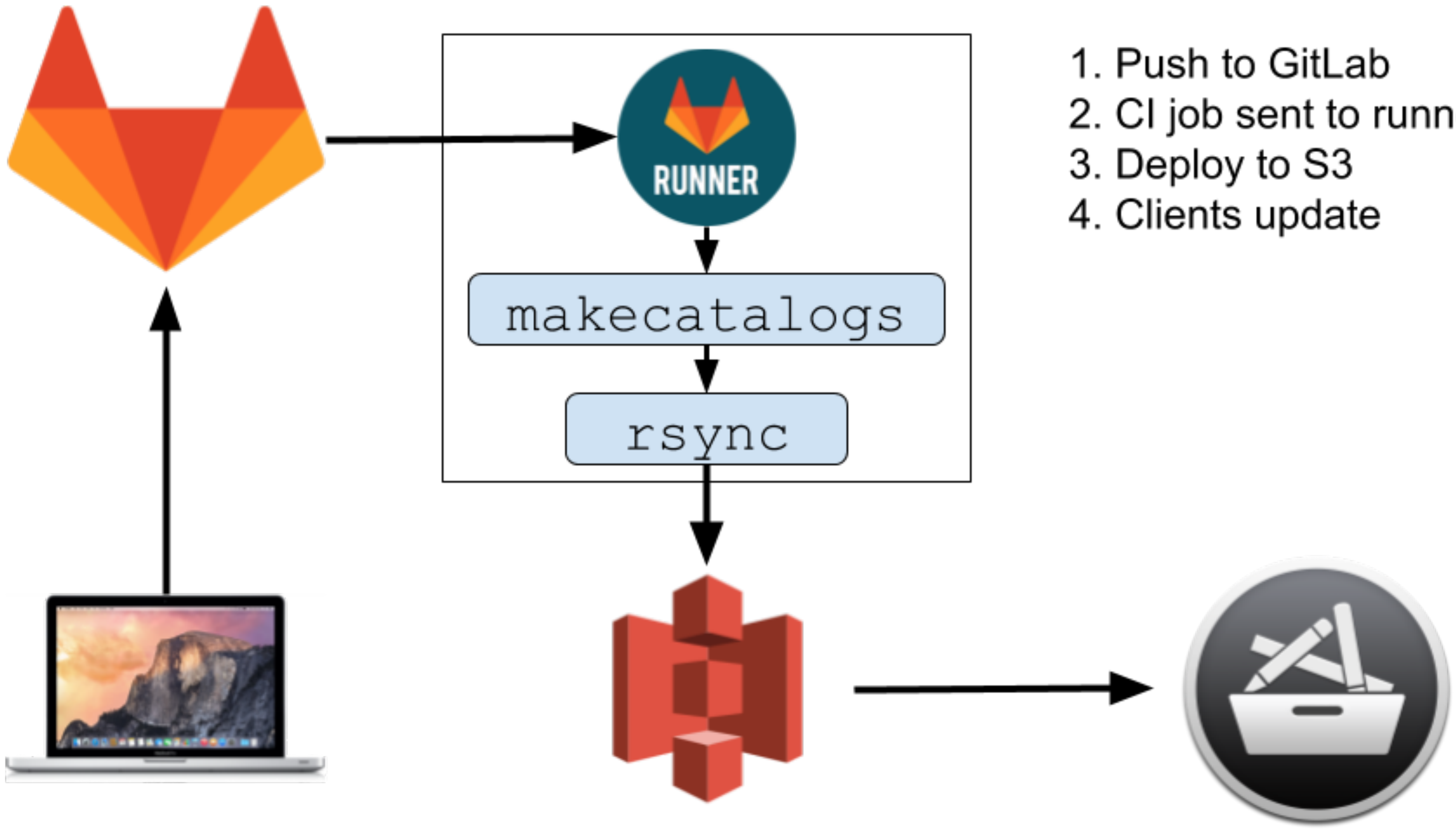
One thing you don't want to do is put secret information, like private keys or API tokens in git.

CI Secret variables allow your CI jobs to access the info securely

Passed as Environment Vars to Runner

Now, you are ready to be taught the new way.

2017: TO THE CLOUD

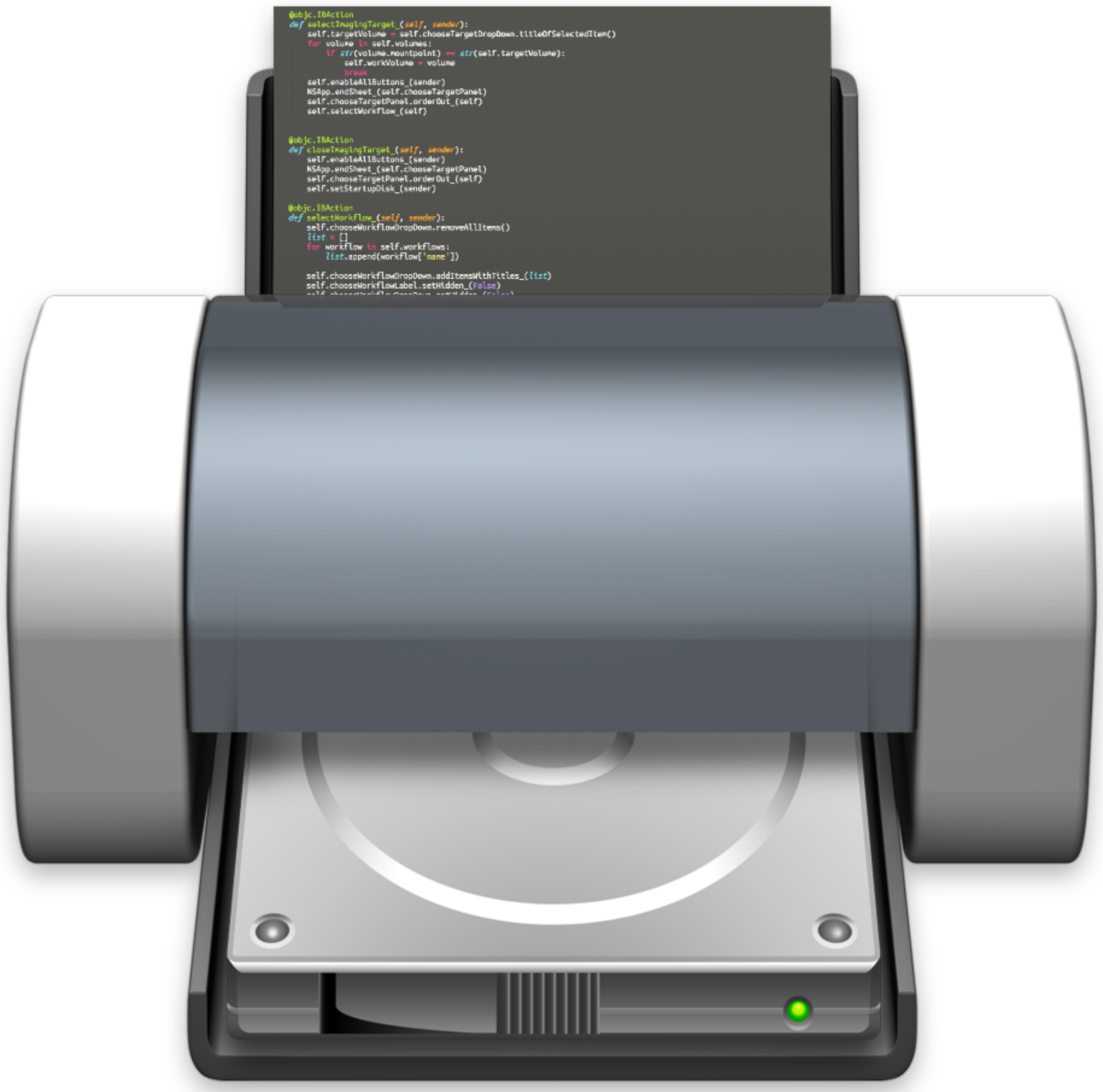


1. Push to GitLab
2. CI job sent to runner
3. Deploy to S3
4. Clients update

So, now I knew about CI.

Also, I figured out it was dumb to keep catalog files in git

And after going to MacDevOps Vancouver in 2016, I wanted to start serving my Munki repo from S3, because it was easy, cheap, and reliable.



and from there I was off to the races

Imagr

updates to my imagr config, packages, and images are all tracked in git, distributed to my site imaging servers

Since the GitLab CI runner supports Mac OS, I set up a runner with an Apple developer certificate that signs all our configuration profiles. I also added a step to lint the plists and validate the contents of some fields, like PayloadOrganization. One time I slipped up and deployed a profile from "Your Org Here" to the whole company.

Next I want to figure out how to make the profiles get added to my Munki repo after being signed.

Next Steps

? 🚗 📦 ?

🚗 📦

AutoPkg is another good use case for CI. I haven't implemented this in GitLab CI myself yet, but Rick Heil has. You put all your overrides in a repo together, and use a nifty script from Facebook CPE to commit updated applications to your Munki repo.

ANSIBLE

Synapse is trying to be an Ansible shop. I hope to accellerate this by automating Ansible Playbook runs through CI.

🐮kins

Mookins

Wes explained this at his talk yesterday, which is an integration between Munki, Oomnitza, his inventory tool, and Jenkins. Jenkins queries the hardware inventory to know who has which computer, and poof suddenly users have manifests that follow them from computer to computer. Watch his talk when it's posted, it was great.

So

So why do all this?

Automatic

Using GitLab and CI makes it much easier to do better IT.

Automation is good, but the more you can control the execution environment, the more reliable it is. With GitLab CI, it's really easy. You set up your runner environment, and then just send your tasks to it.

Accountable

You get to see exactly what changed, who changed it, and when. You get your git history, as well as the execution transcripts of every job sent to CI.

Collaborative

"Hey, are you on the server?" Nobody likes that question. By routing our workflows through GitLab, anybody can jump in the process. If our work conflicts, we find out before it gets into production.

References

- PSU MacAdmins 2017
  - Tom Bridge - Munki Mistakes Made Right
  - Lucas Hall - Managing MacOS without MacOS (almost)
  - Wesley Whetstone - Continuous Integration: An automation framework for Mac Admins.
  - Rick Heil - Advanced Munki Infrastructure: Moving to Cloud Services
- MacDevOps:YVR 2016
  - Tim Sutton on Jenkins CI
  - Wade Robson on Munki & S3

Thanks!

Twitter: @macjustice
MacAdmins Slack: macjustice
Wherever: macjustice

Q & A