
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

Keeping Helm Reliable And Usable

Matt Farina, SUSE
Karena Angell, Red Hat
George Jenkins, Bloomberg
Ian Zink, Independent


KubeCon | CloudNativeCon

North America 2023

Keeping Helm Reliable And Usable

Stable

Matt Farina, SUSE
Karena Angell, Red Hat
George Jenkins, Bloomberg
Ian Zink, Independent

Helm is a foundational part of Kubernetes software delivery. Deploying mission critical software for many projects and users. Keeping Helm stable while also improving usability is an important goal of the Helm project as any change can affect a large portion of the Kubernetes community. In this session, we will review how the Helm project prioritizes users, enforces a robust change management process, security, testing, and more while still layering in new features to keep its ecosystem reliable, compelling, and usable for you.

Notes

- What is reliability (stability) and usability
- Why Helm is concerned with these things
- Helm has staying power: designed for longevity
- reliability / stability
  - semver
  - Governance
  - testing
  - pace of change
    - maturity
    - independence
  - community
  - trust
- Usable
  - Scope of Helm (what's in and what's out)
- Metrics on how Helm is doing


KubeCon | CloudNativeCon

North America 2023

What Do We Mean By Reliable, Stable, and Usable?

Define: Reliable

“deserving trust; dependable”

“a machine, piece of equipment, or system that is reliable always works well without breaking down”

– Cambridge Dictionary

Define: Reliable

"Software reliability is the ***probability of the software causing a system failure*** over some specified operating time. Software does not fail due to wear out but does fail due to faulty functionality, timing, sequencing, data, and exception handling. Software fails as a function of operating time as opposed to calendar time."

– Wikipedia

Define: Stable

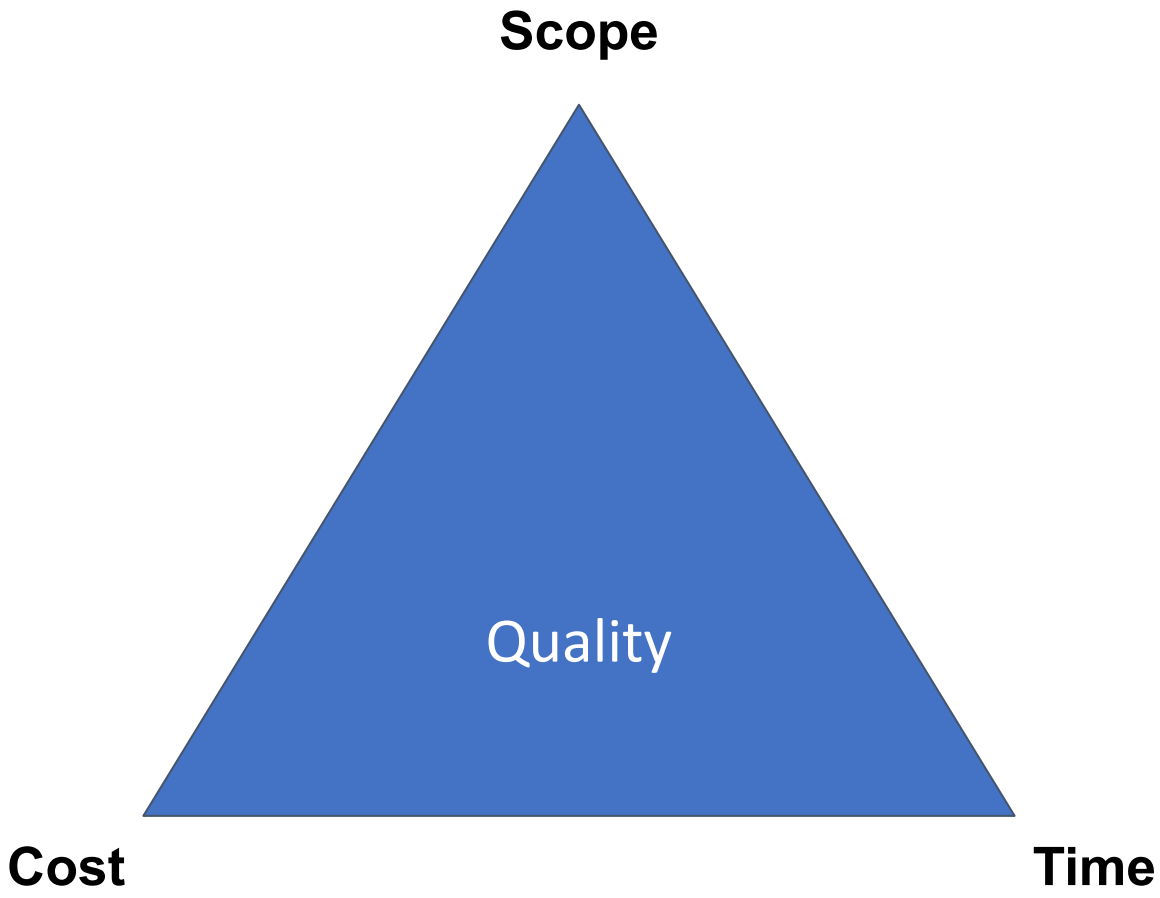
“firmly established”

“not changing or fluctuating”

“steady in purpose”

– Merriam-Webster Dictionary

Stability vs. Features



Define: Usable

"Useful = usability + utility"

– Jakob Nielsen (via the Nielsen Norman Group)

Define: Usability

"Usability = how easy & pleasant these features are to use."

– Jakob Nielsen

Define: Usability

“Usability is a measure of how well a specific user in a specific context can use a product/design to achieve a defined goal effectively, efficiently and satisfactorily.”

– Interaction Design Foundation


KubeCon | CloudNativeCon

North America 2023

Why Is Helm Concerned With These Things?

Highlights: Downloads Last Month

~2.2 M

Latest Version Downloads

> 11 M

Total Downloads

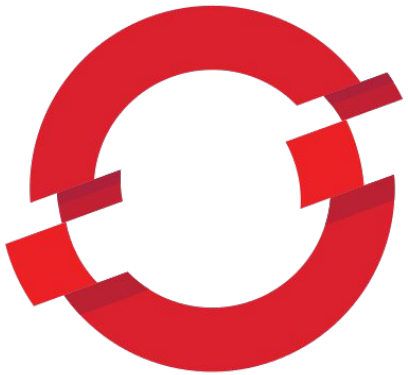
> 151 TB

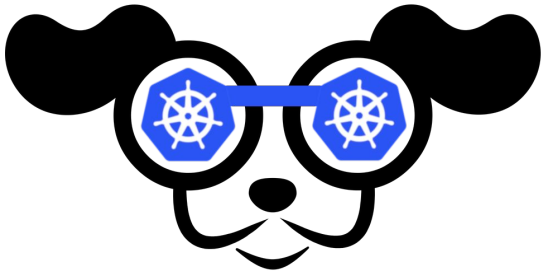
Bandwidth Transferred

Highlights: Old Version DLs Last Month

> 440k	> 470k	> 350k
3.9.0	3.2.1	2.17.0

Applications Using Helm SDK



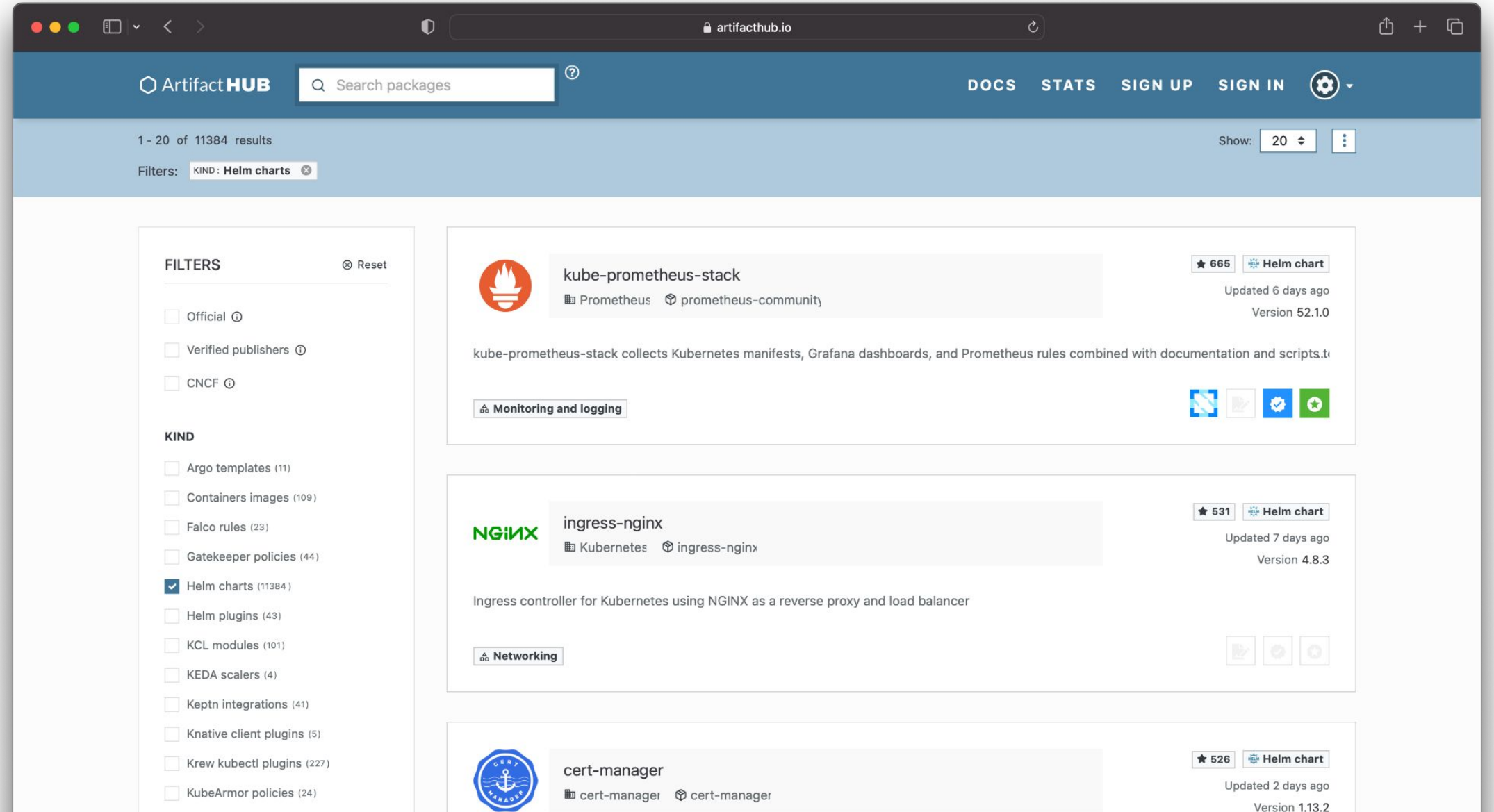


Scripts Using Helm CLI

10s of thousands of shell scripts on GitHub

Public Charts

KubeCon CloudNativeCon North America 2023



Graduated

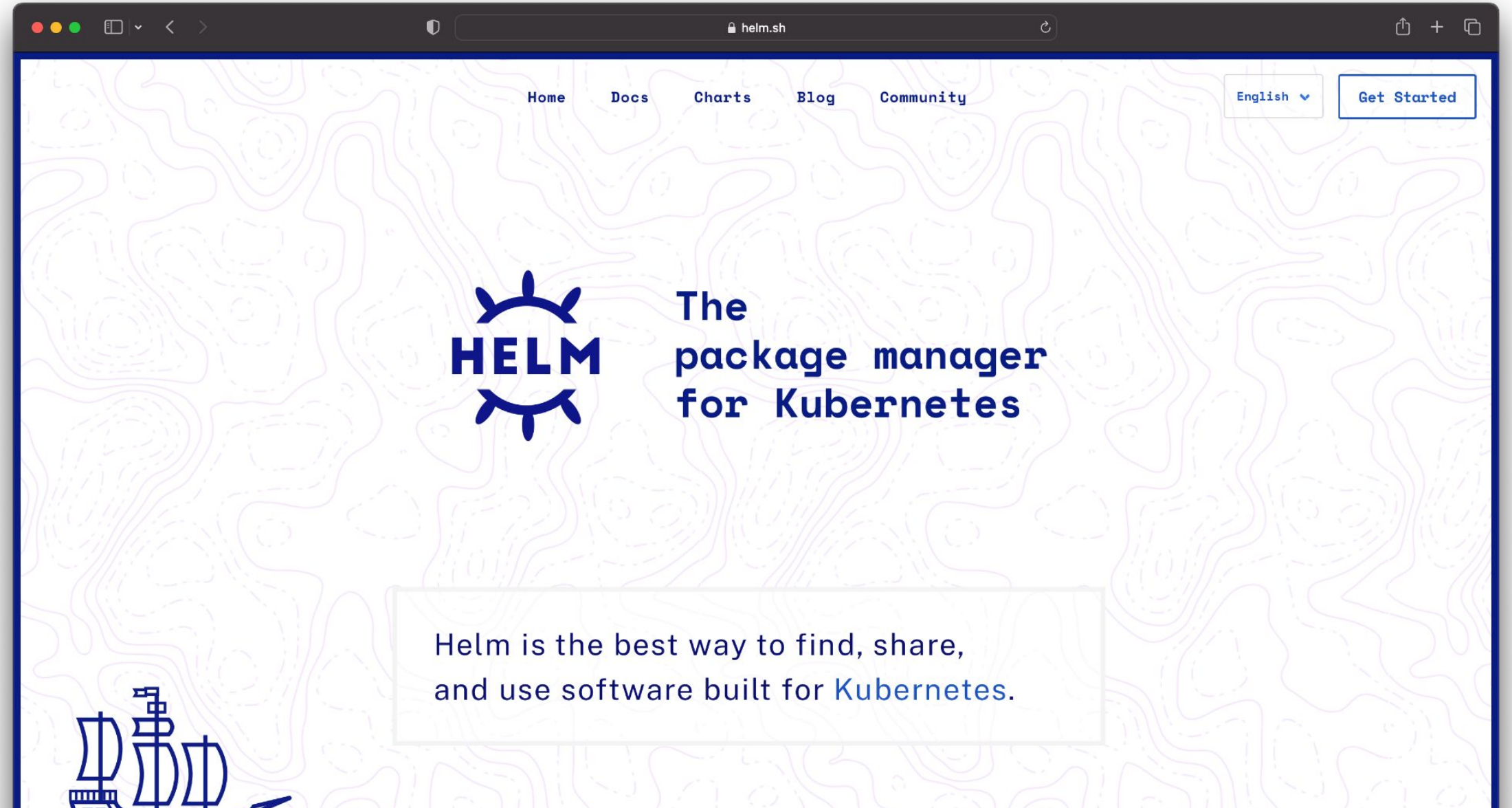
Helm is widely used and expected to be mature software.


KubeCon | CloudNativeCon

North America 2023

How Helm Pulls This Off: Scope

What Is Helm?


helm.sh
Home
Docs
Charts
Blog
Community
English
Get Started
HELM
The package manager for Kubernetes
Helm is the best way to find, share, and use software built for Kubernetes.

Define: Package Manager

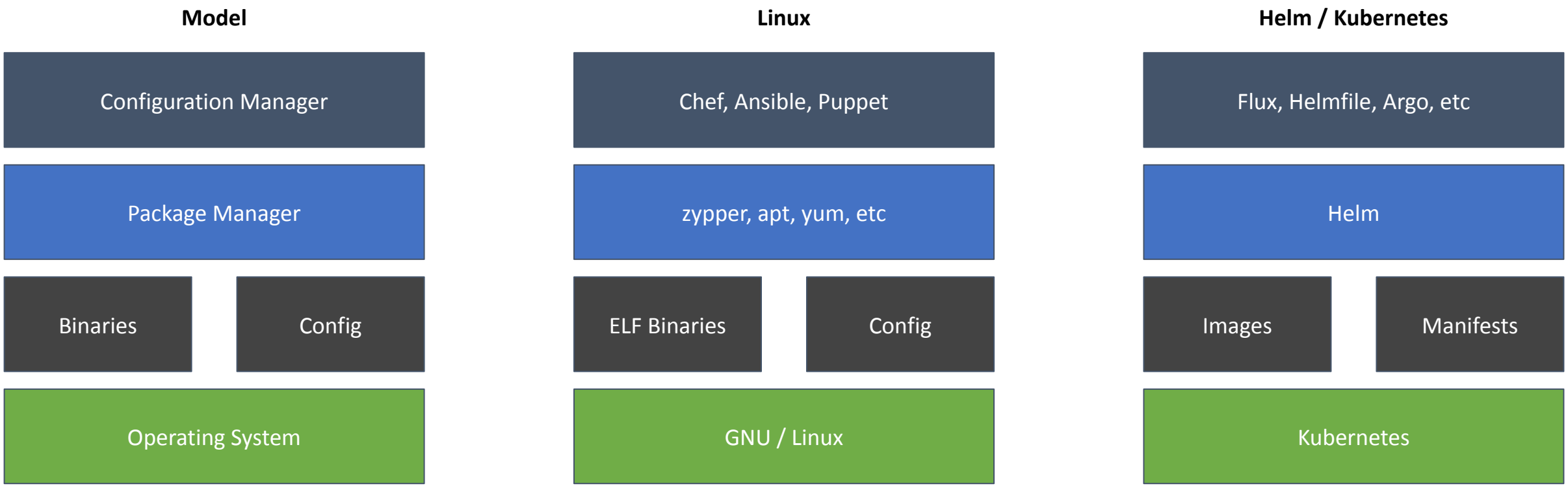
"A package manager or package-management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer in a consistent manner."

– Wikipedia: Package Manager

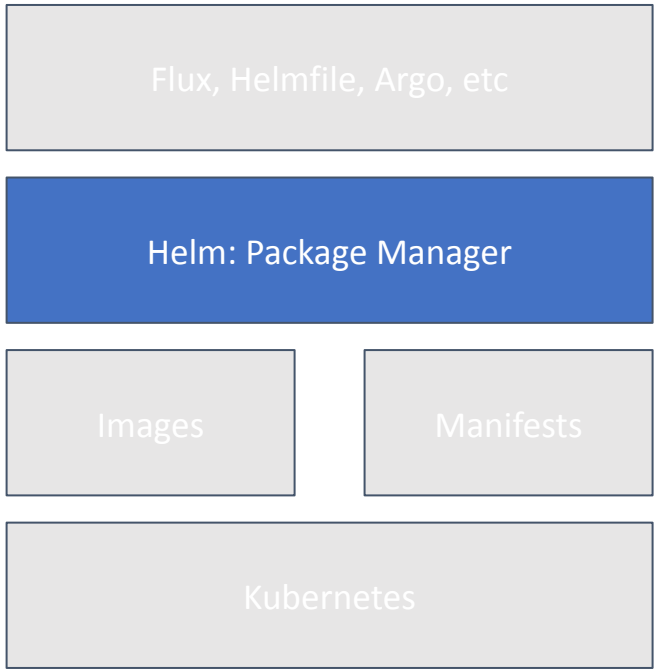
Define: Package Manager

Package Manager != Configuration Manager

Helm And Other Tools



Managing Scope



What features, should a package manager accept?

Managing Scope

Helm: Package Manager

What features, should a package manager accept?

Where is the line between package management and higher level orchestration?

Prioritized User Profiles and Roles

We have User Profiles/Roles (in priority order):

1. Application Operator
2. Application Distributor
3. Application Developer
4. Supporting Tool Developer
5. Helm Developer

Out of scope:

- Cluster Operator

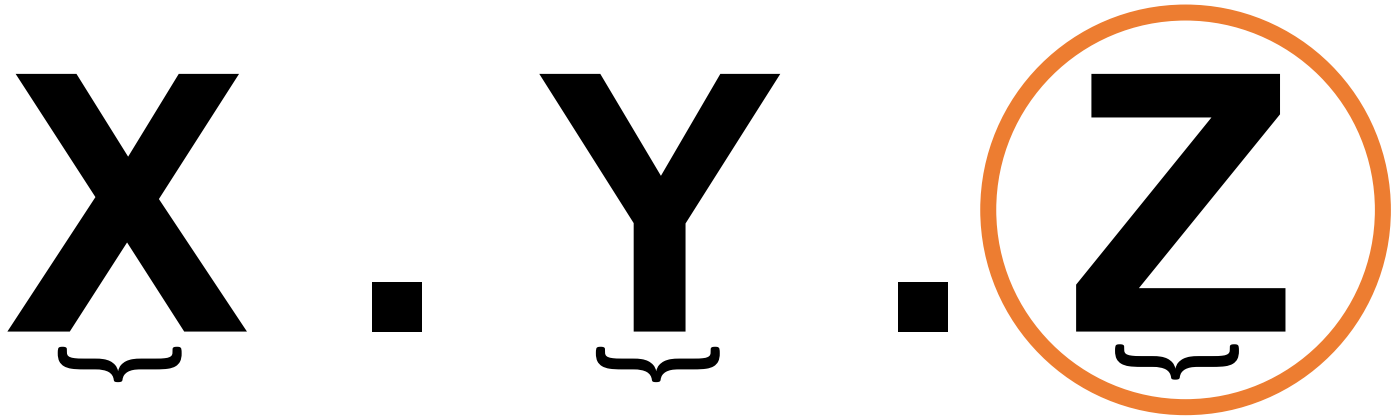

KubeCon | CloudNativeCon

North America 2023

How Helm Pulls This Off: Versioning & Policies

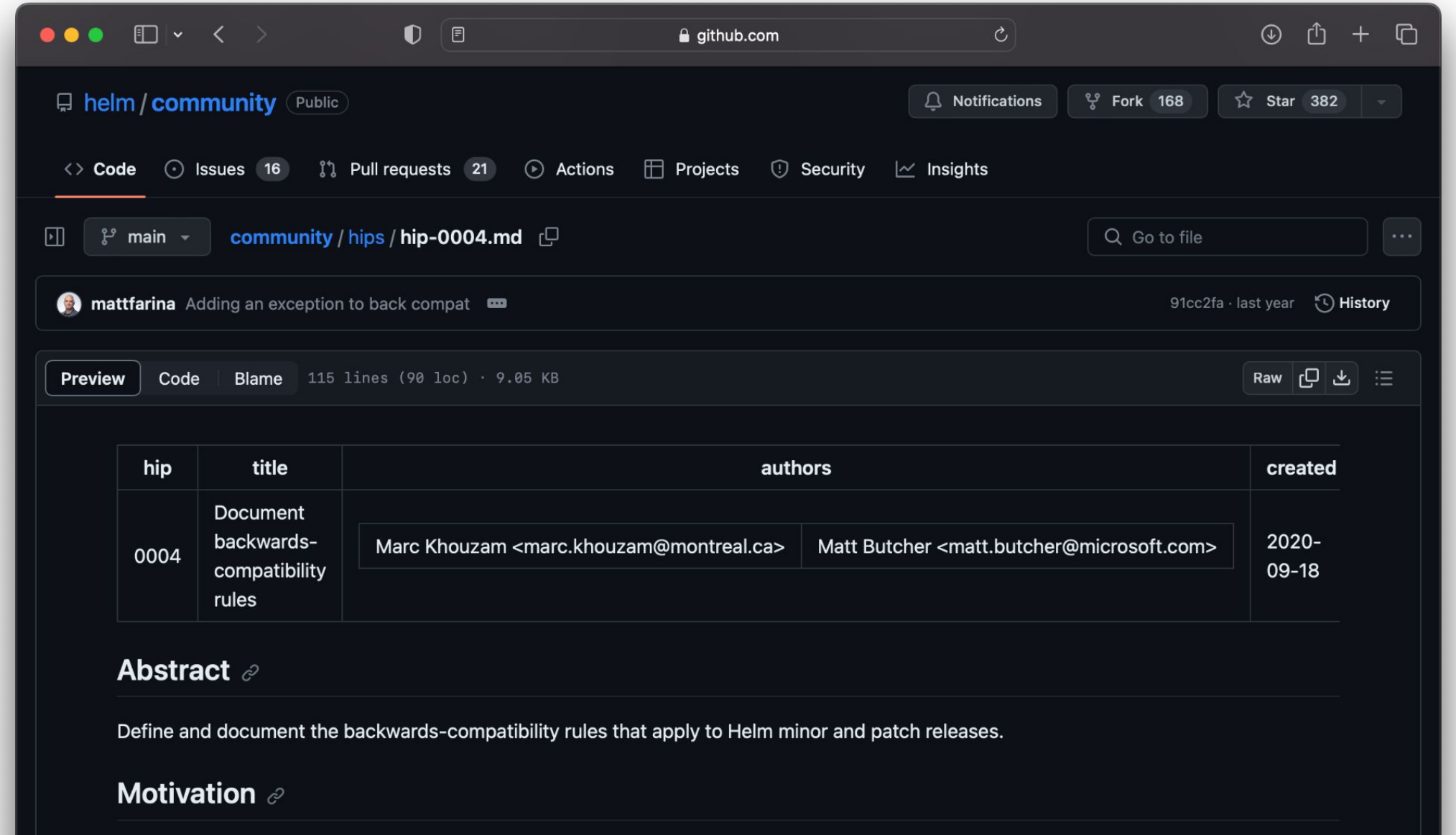
Semantic Versioning

Semantic Versioning



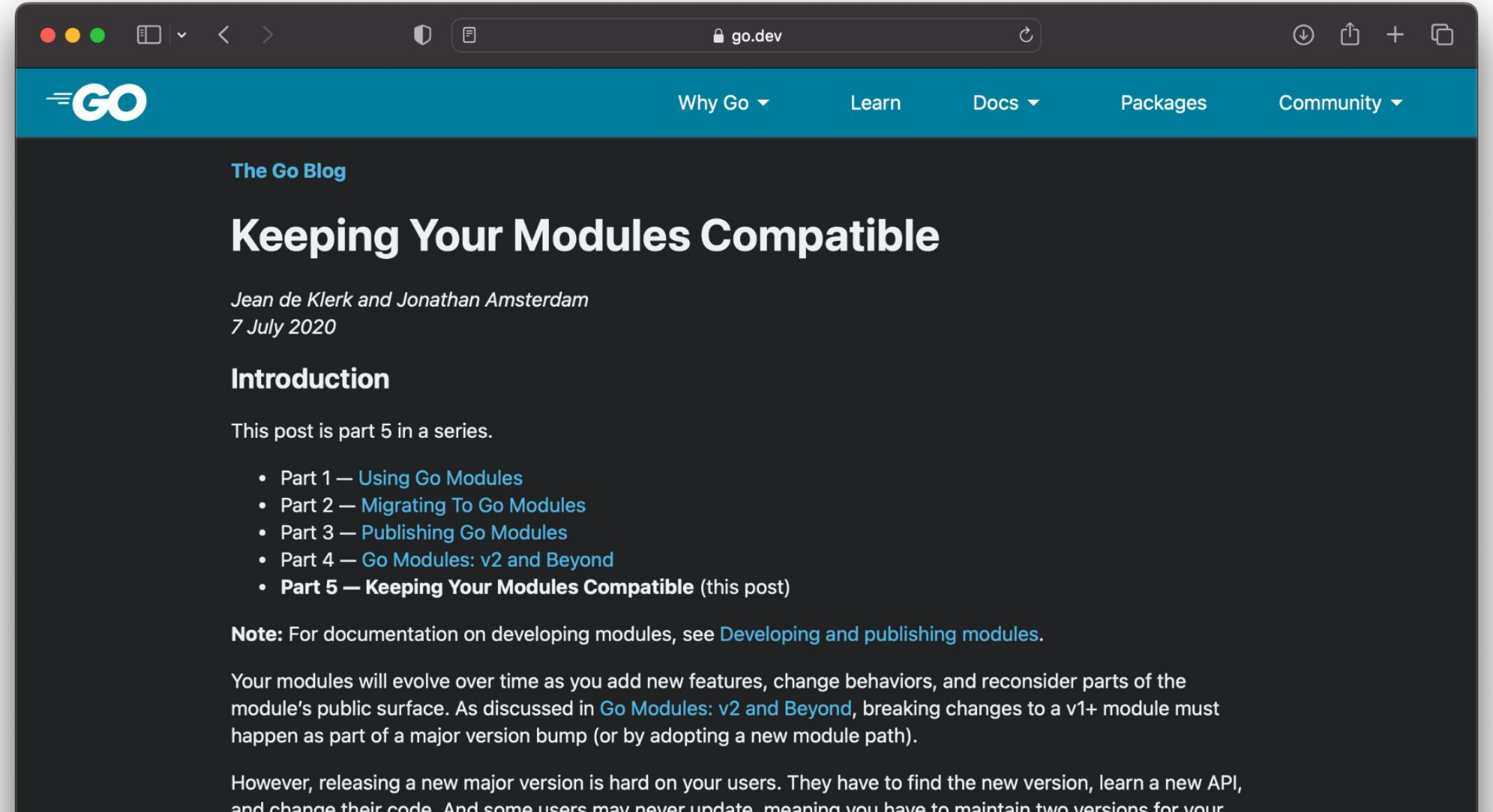
Backwards Compatibility

KubeCon CloudNativeCon North America 2023

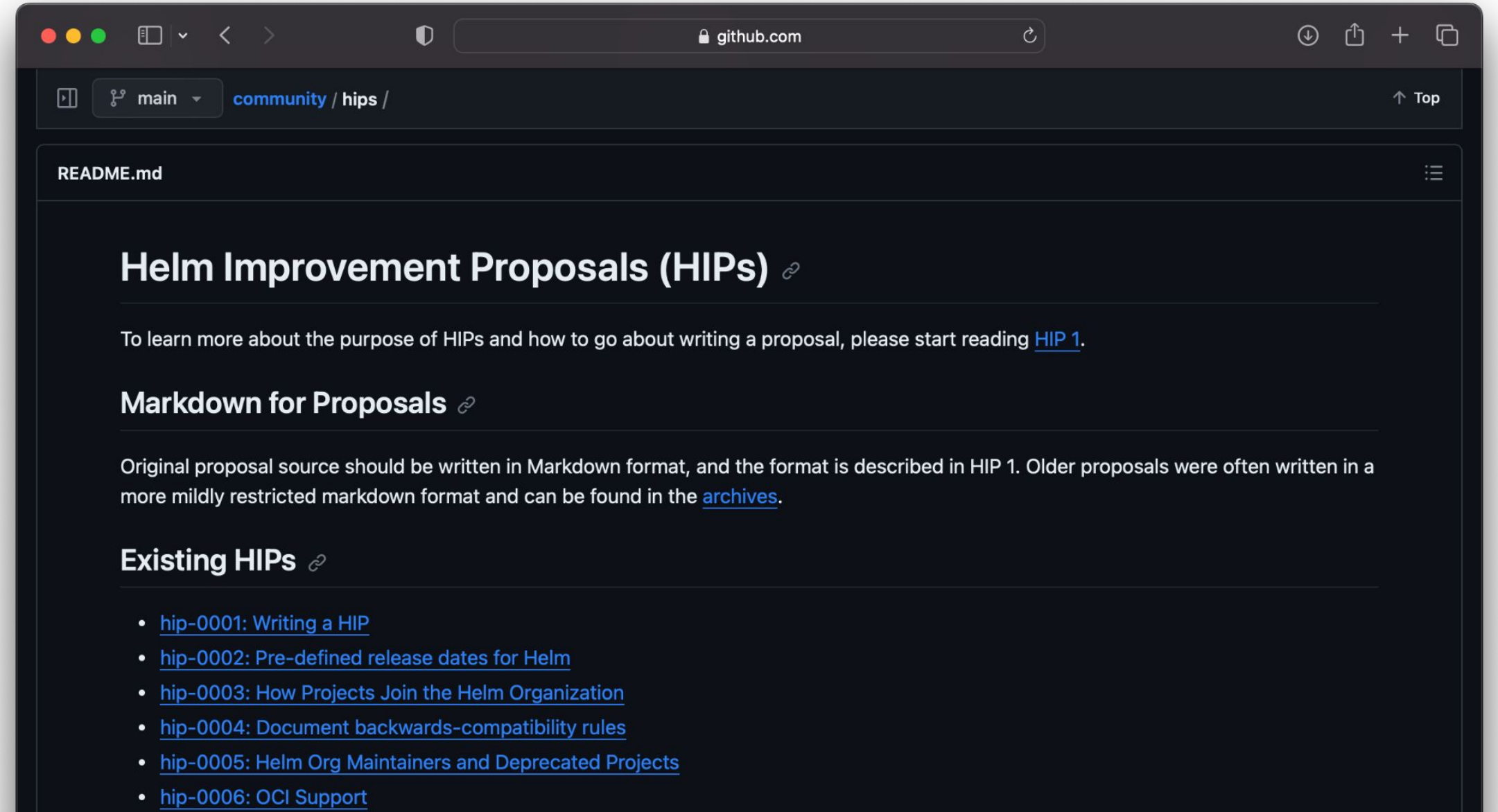


Go Module Compatibility

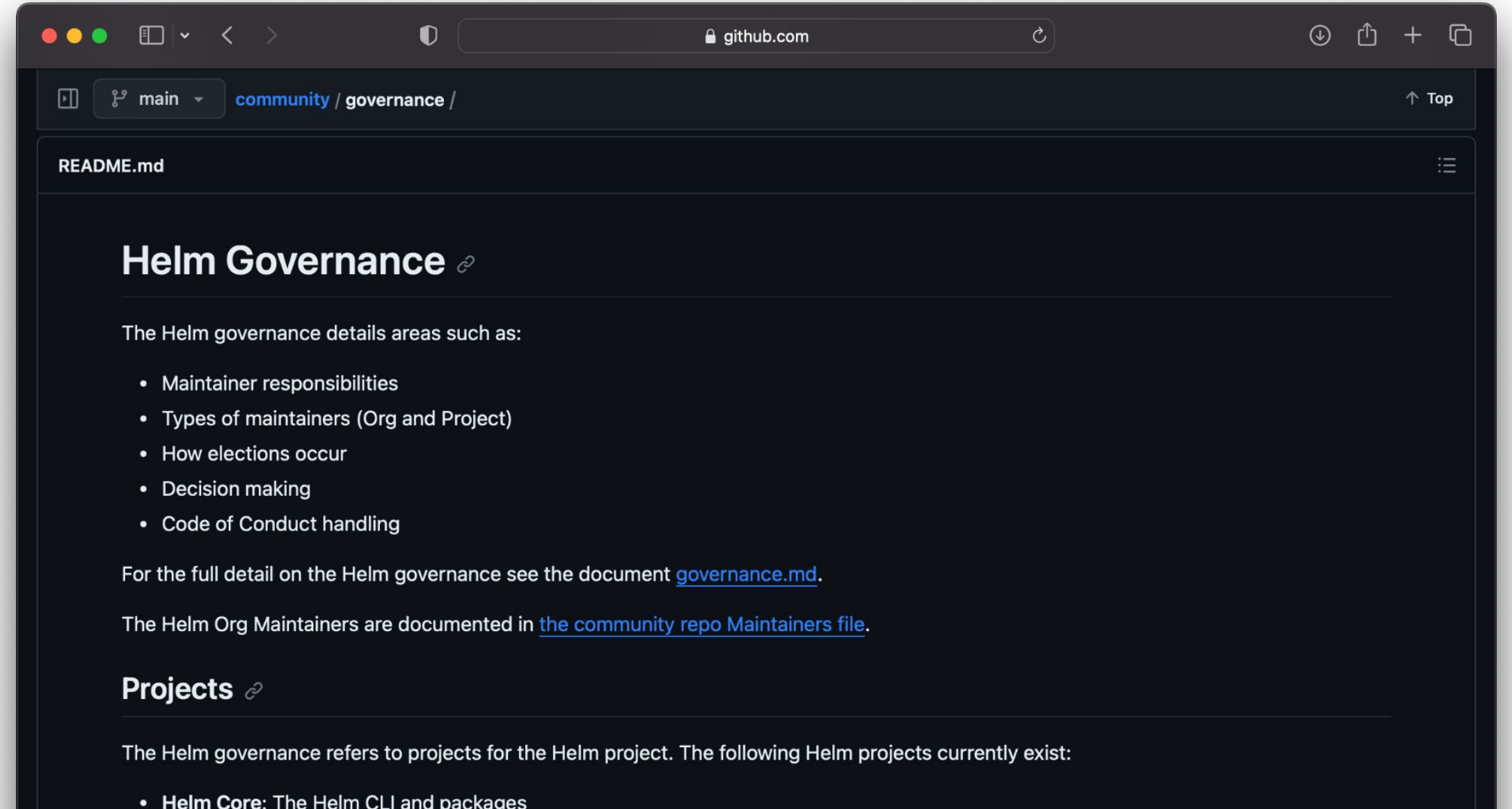
KubeCon CloudNativeCon North America 2023



Policies

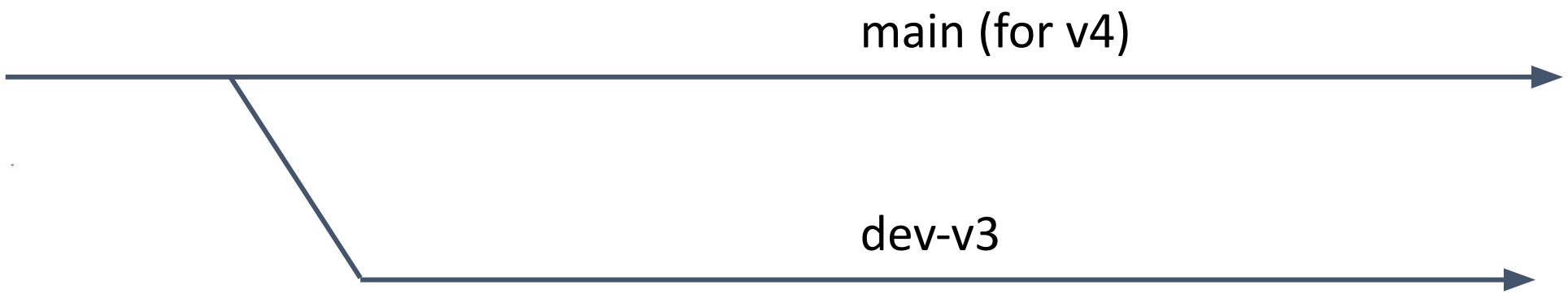

github.com
main
community / hips /
Top
README.md
Helm Improvement Proposals (HIPs)
To learn more about the purpose of HIPs and how to go about writing a proposal, please start reading HIP 1.
Markdown for Proposals
Original proposal source should be written in Markdown format, and the format is described in HIP 1. Older proposals were often written in a more mildly restricted markdown format and can be found in the archives.
Existing HIPs
hip-0001: Writing a HIP
hip-0002: Pre-defined release dates for Helm
hip-0003: How Projects Join the Helm Organization
hip-0004: Document backwards-compatibility rules
hip-0005: Helm Org Maintainers and Deprecated Projects
hip-0006: OCI Support

Governance

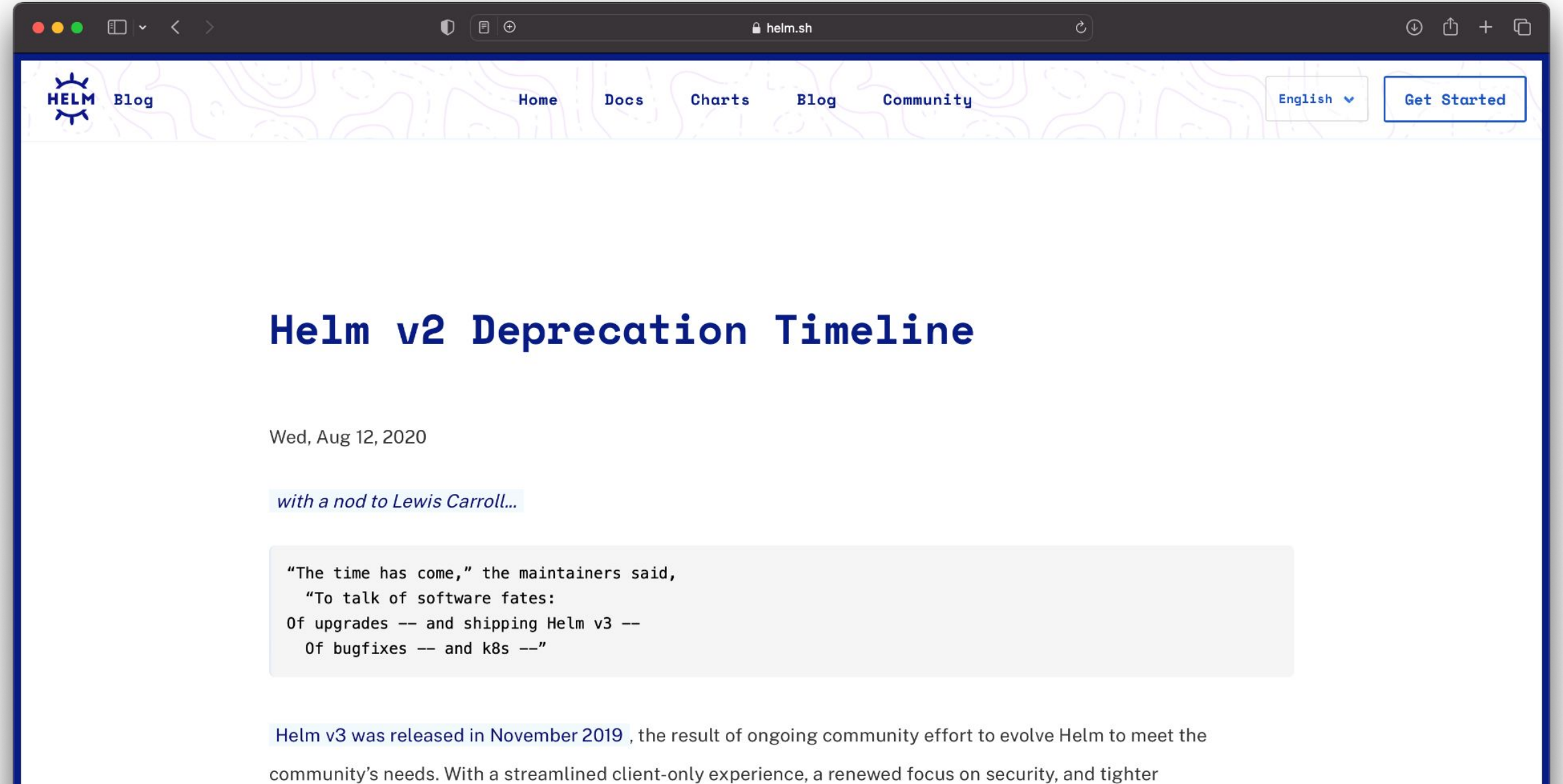

github.com
main
community / governance /
Top
README.md
Helm Governance
The Helm governance details areas such as:
Maintainer responsibilities
Types of maintainers (Org and Project)
How elections occur
Decision making
Code of Conduct handling
For the full detail on the Helm governance see the document governance.md.
The Helm Org Maintainers are documented in the community repo Maintainers file.
Projects
The Helm governance refers to projects for the Helm project. The following Helm projects currently exist:
Helm Core: The Helm CLI and packages

Helm v4

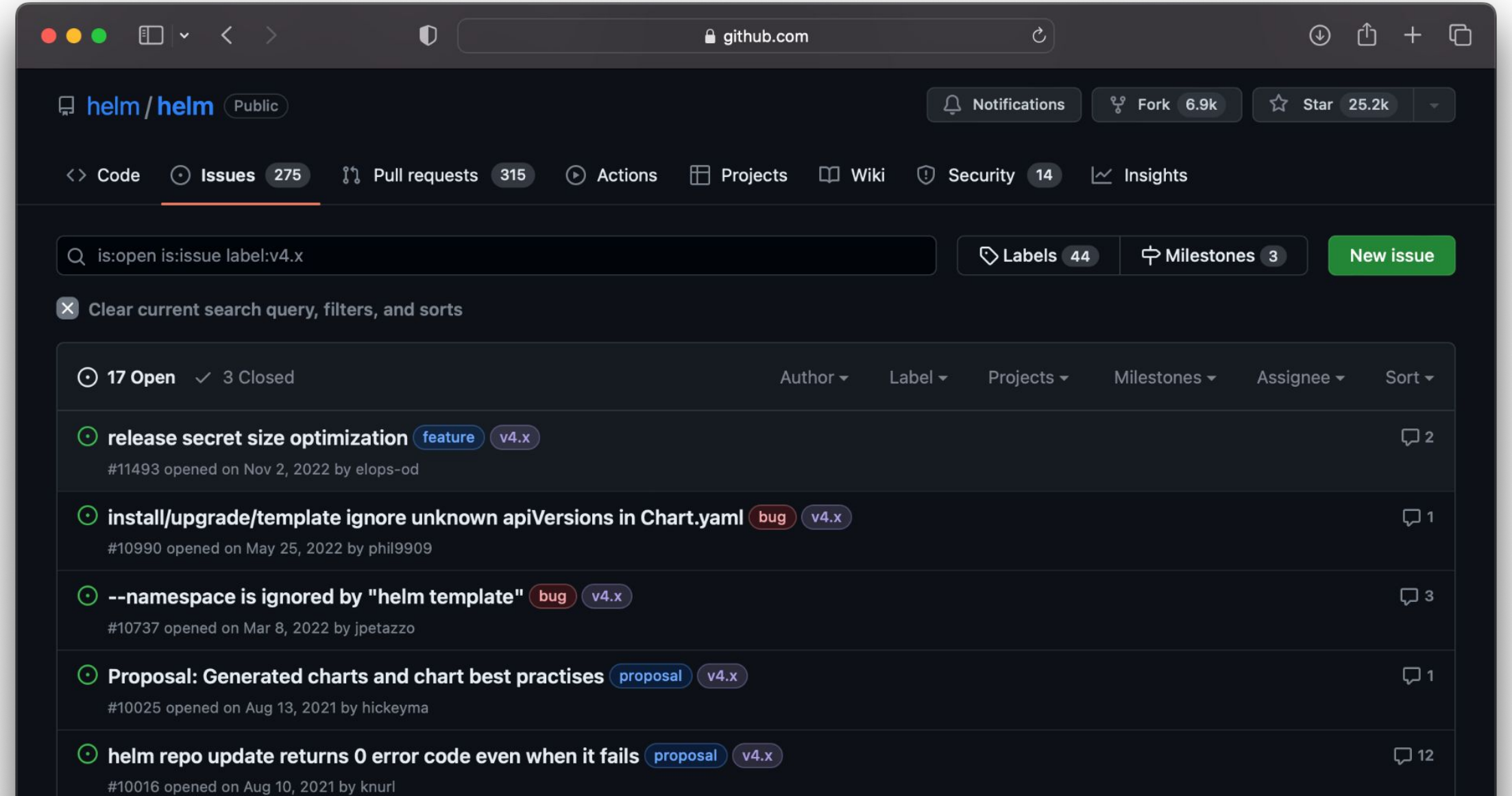
How We Work



Helm 3 support plan



What Do We Change?

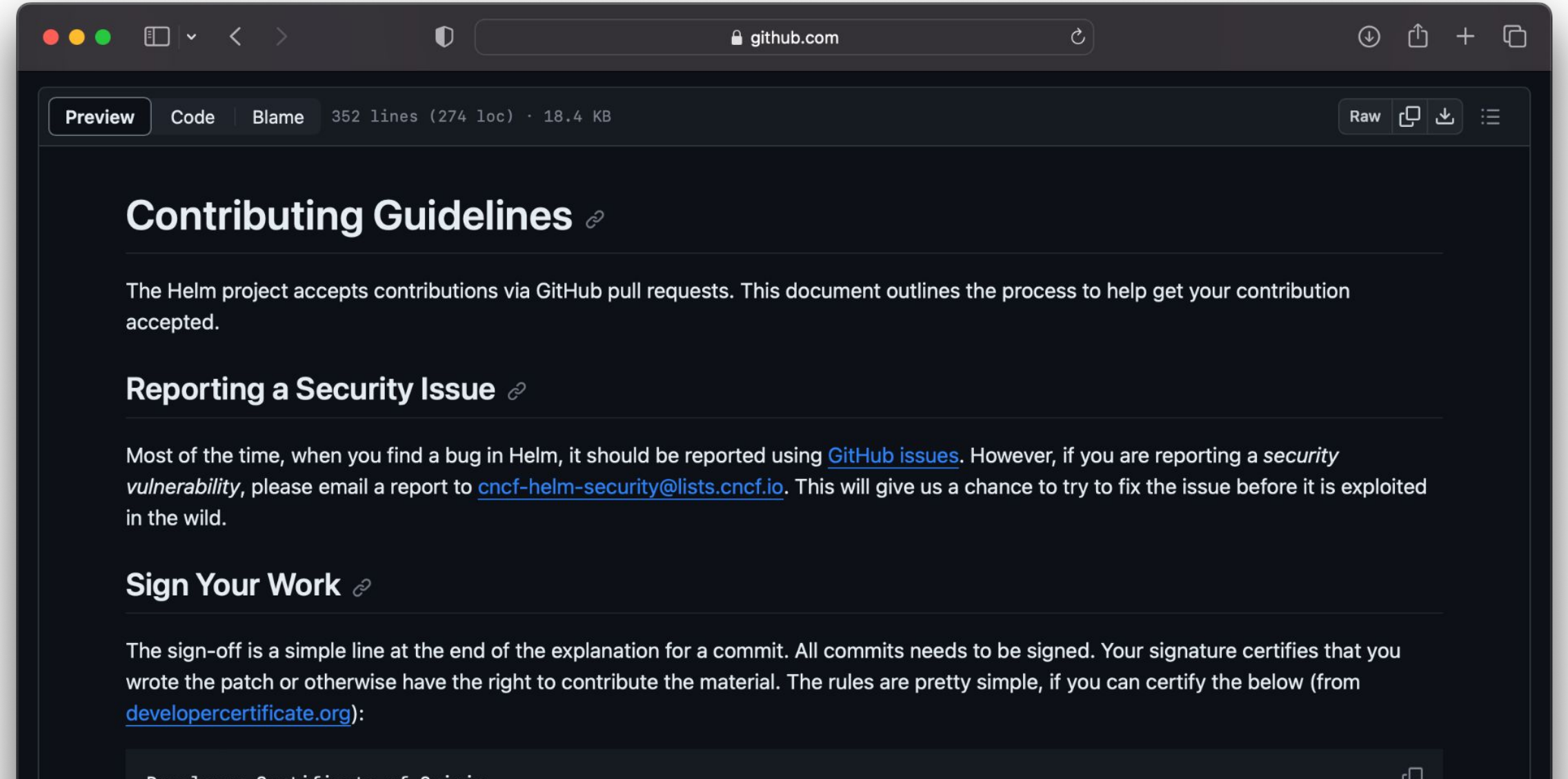


Not Breaking Charts

Chart.yaml:

```
apiVersion: v2
…
```

Want To Help?


github.com
Preview
Code
Blame
352 lines (274 loc) · 18.4 KB
Raw
Contributing Guidelines
The Helm project accepts contributions via GitHub pull requests. This document outlines the process to help get your contribution accepted.
Reporting a Security Issue
Most of the time, when you find a bug in Helm, it should be reported using GitHub issues. However, if you are reporting a security vulnerability, please email a report to cncf-helm-security@lists.cncf.io. This will give us a chance to try to fix the issue before it is exploited in the wild.
Sign Your Work
The sign-off is a simple line at the end of the explanation for a commit. All commits needs to be signed. Your signature certifies that you wrote the patch or otherwise have the right to contribute the material. The rules are pretty simple, if you can certify the below (from developercertificate.org):

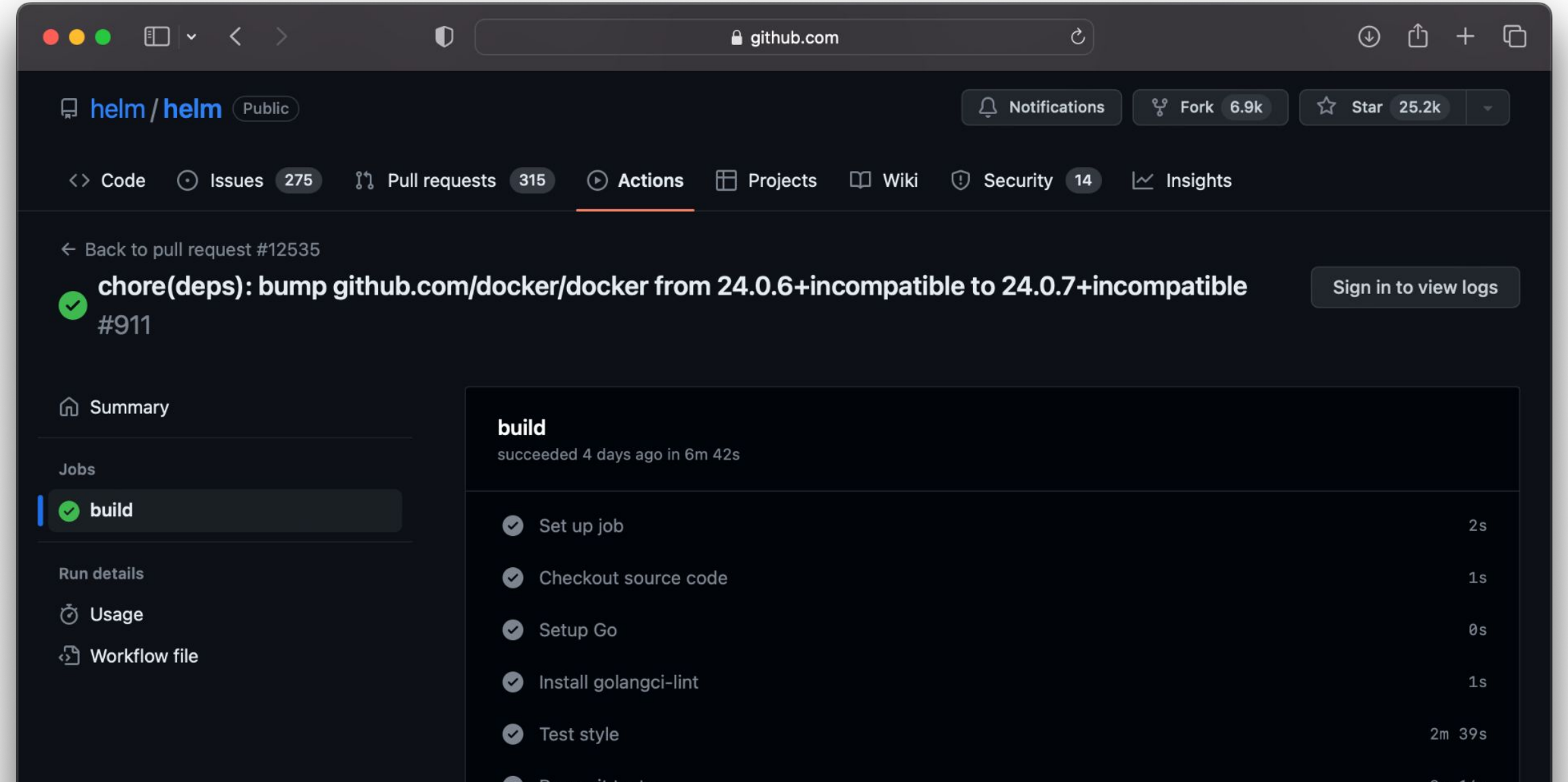

KubeCon | CloudNativeCon

North America 2023

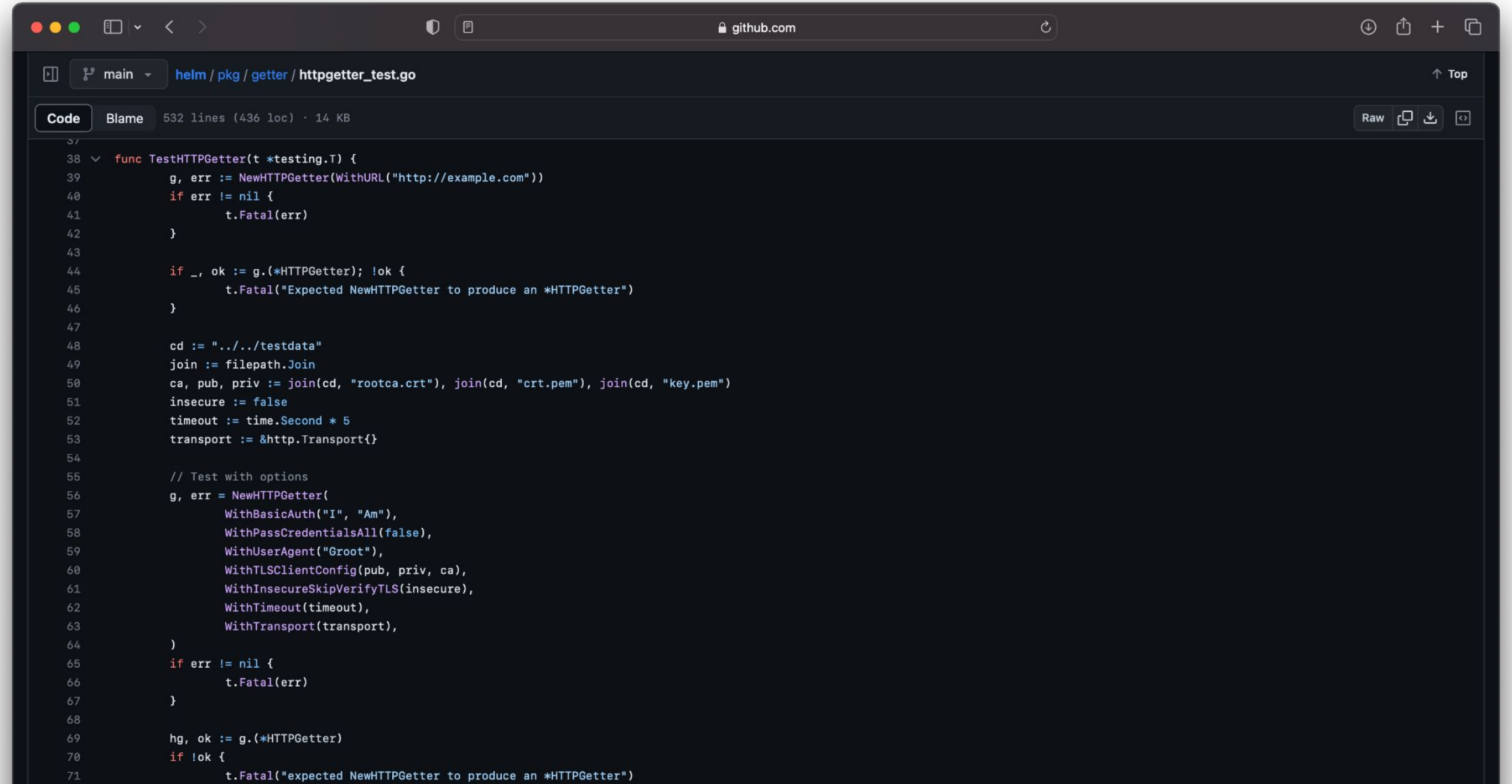
How Helm Pulls This Off: Testing

Tests In Go

KubeCon CloudNativeCon North America 2023

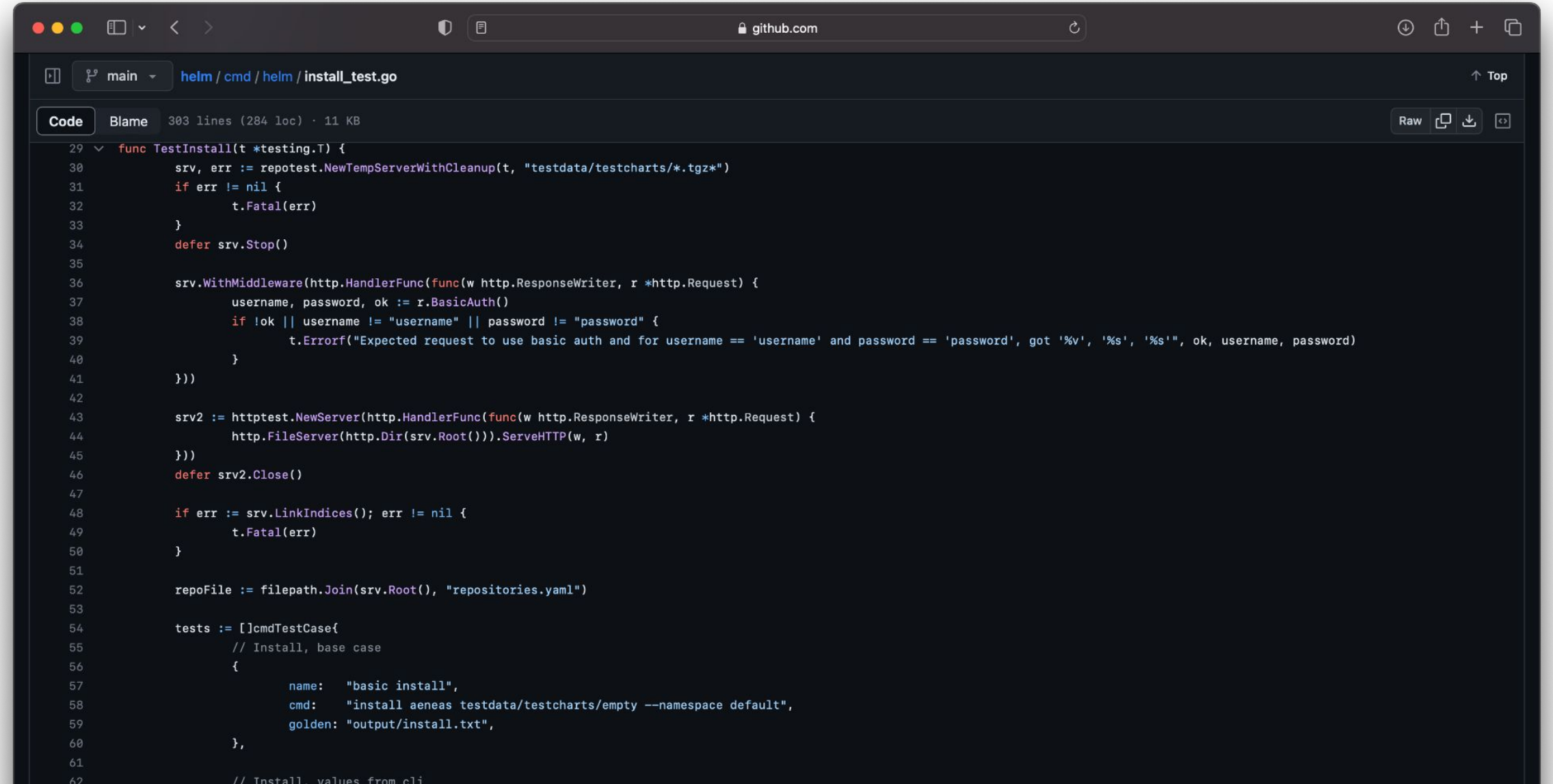


Unit and API



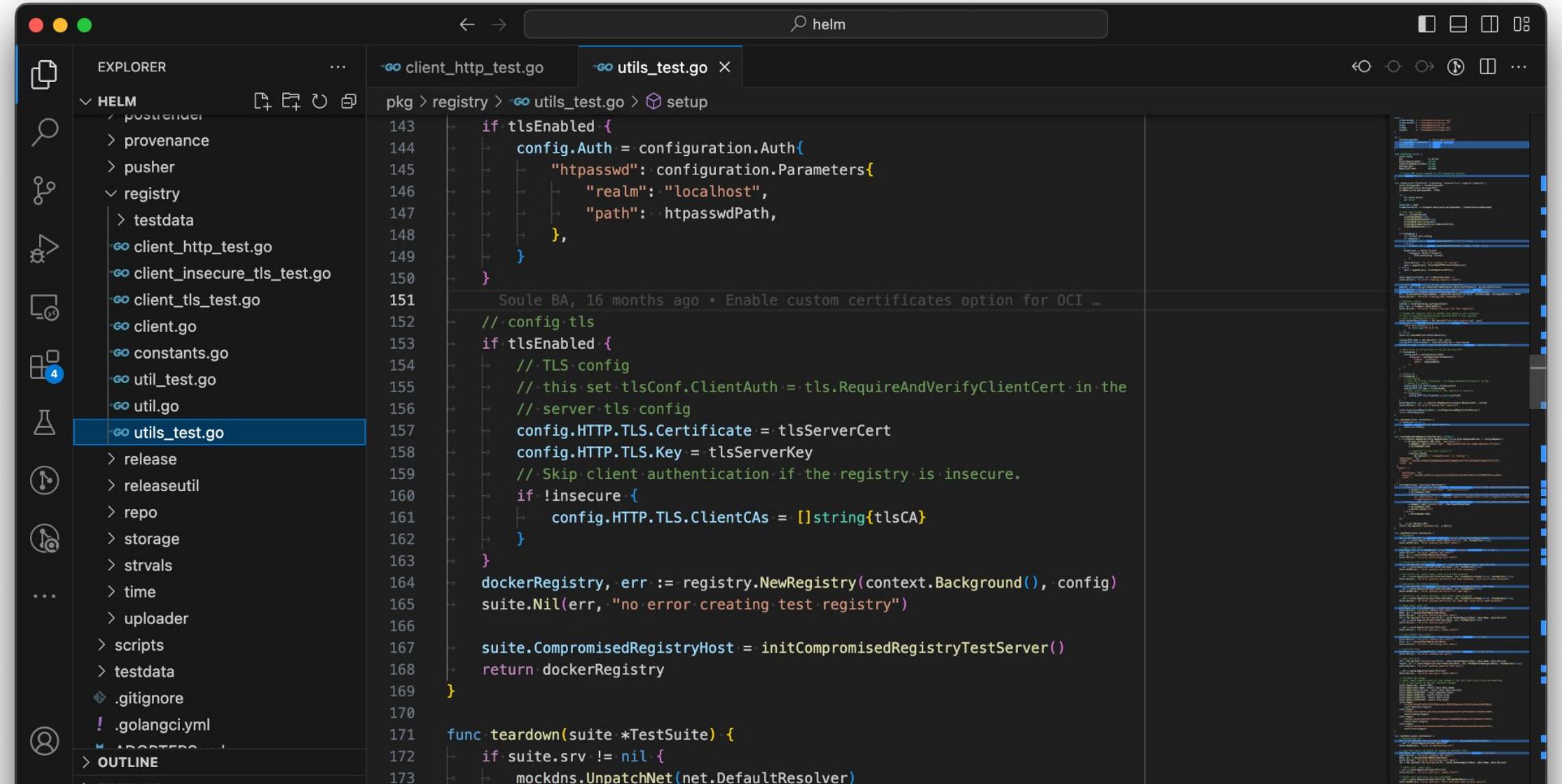

```go
func TestHTTPGetter(t *testing.T) {
	g, err := NewHTTPGetter(WithURL("http://example.com"))
	if err != nil {
		t.Fatal(err)
	}

	if _, ok := g.(*HTTPGetter); !ok {
		t.Fatal("Expected NewHTTPGetter to produce an *HTTPGetter")
	}

	cd := "../../testdata"
	join := filepath.Join
	ca, pub, priv := join(cd, "rootca.crt"), join(cd, "crt.pem"), join(cd, "key.pem")
	insecure := false
	timeout := time.Second * 5
	transport := &http.Transport{}

	// Test with options
	g, err = NewHTTPGetter(
		WithBasicAuth("I", "Am"),
		WithPassCredentialsAll(false),
		WithUserAgent("Groot"),
		WithTLSClientConfig(pub, priv, ca),
		WithInsecureSkipVerifyTLS(insecure),
		WithTimeout(timeout),
		WithTransport(transport),
	)
	if err != nil {
		t.Fatal(err)
	}

	hg, ok := g.(*HTTPGetter)
	if !ok {
		t.Fatal("expected NewHTTPGetter to produce an *HTTPGetter")
```

Testing CLI Output

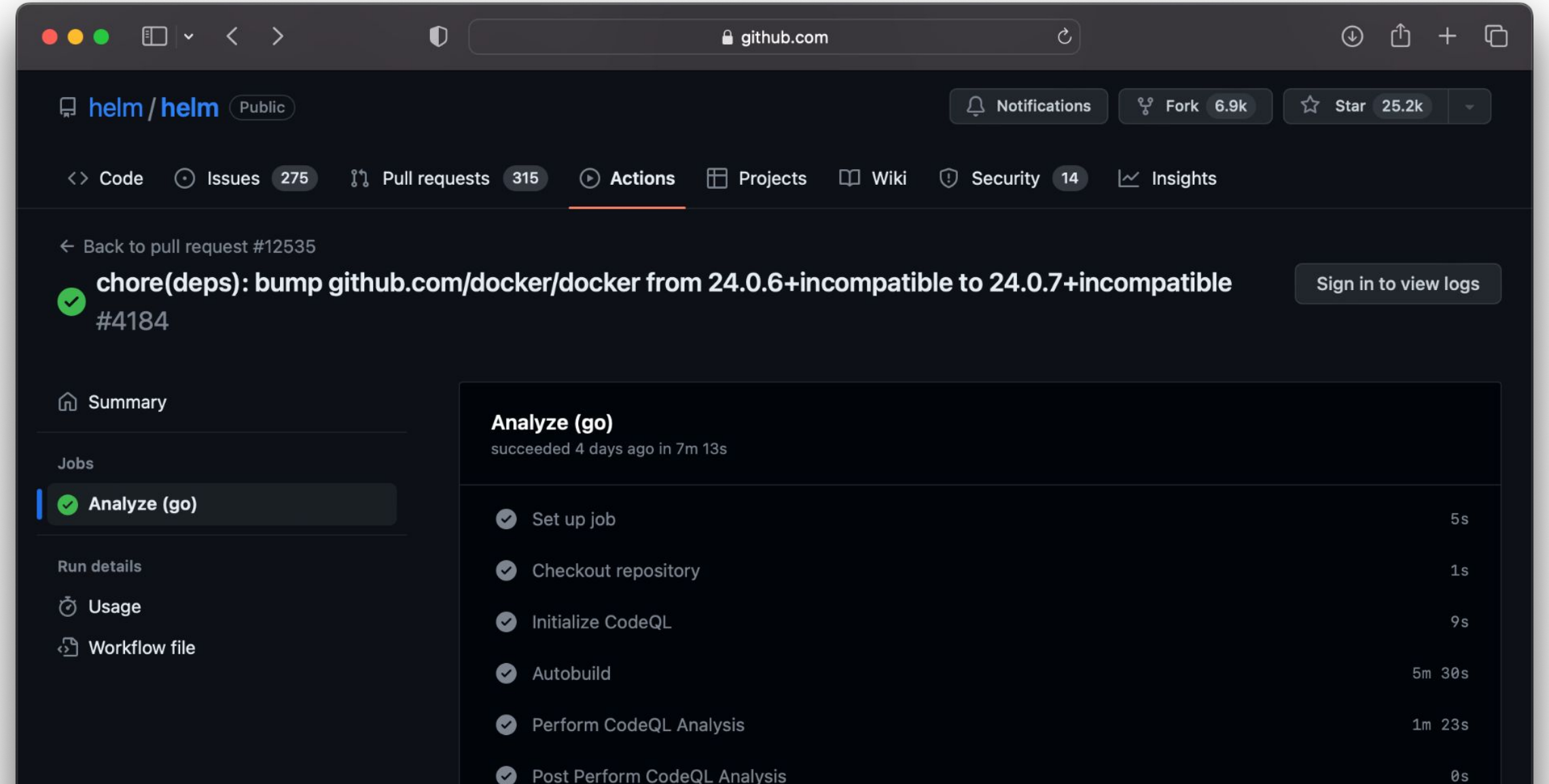



```go
func TestInstall(t *testing.T) {
	srv, err := repotest.NewTempServerWithCleanup(t, "testdata/testcharts/*.tgz*")
	if err != nil {
		t.Fatal(err)
	}
	defer srv.Stop()

	srv.WithMiddleware(http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
		username, password, ok := r.BasicAuth()
		if !ok || username != "username" || password != "password" {
			t.Errorf("Expected request to use basic auth and for username == 'username' and password == 'password', got '%v', '%s', '%s'", ok, username, password)
		}
	}))

	srv2 := httptest.NewServer(http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
		http.FileServer(http.Dir(srv.Root())).ServeHTTP(w, r)
	}))
	defer srv2.Close()

	if err := srv.LinkIndices(); err != nil {
		t.Fatal(err)
	}

	repoFile := filepath.Join(srv.Root(), "repositories.yaml")

	tests := []cmdTestCase{
		// Install, base case
		{
			name:   "basic install",
			cmd:    "install aeneas testdata/testcharts/empty --namespace default",
			golden: "output/install.txt",
		},

		// Install, values from cli
```

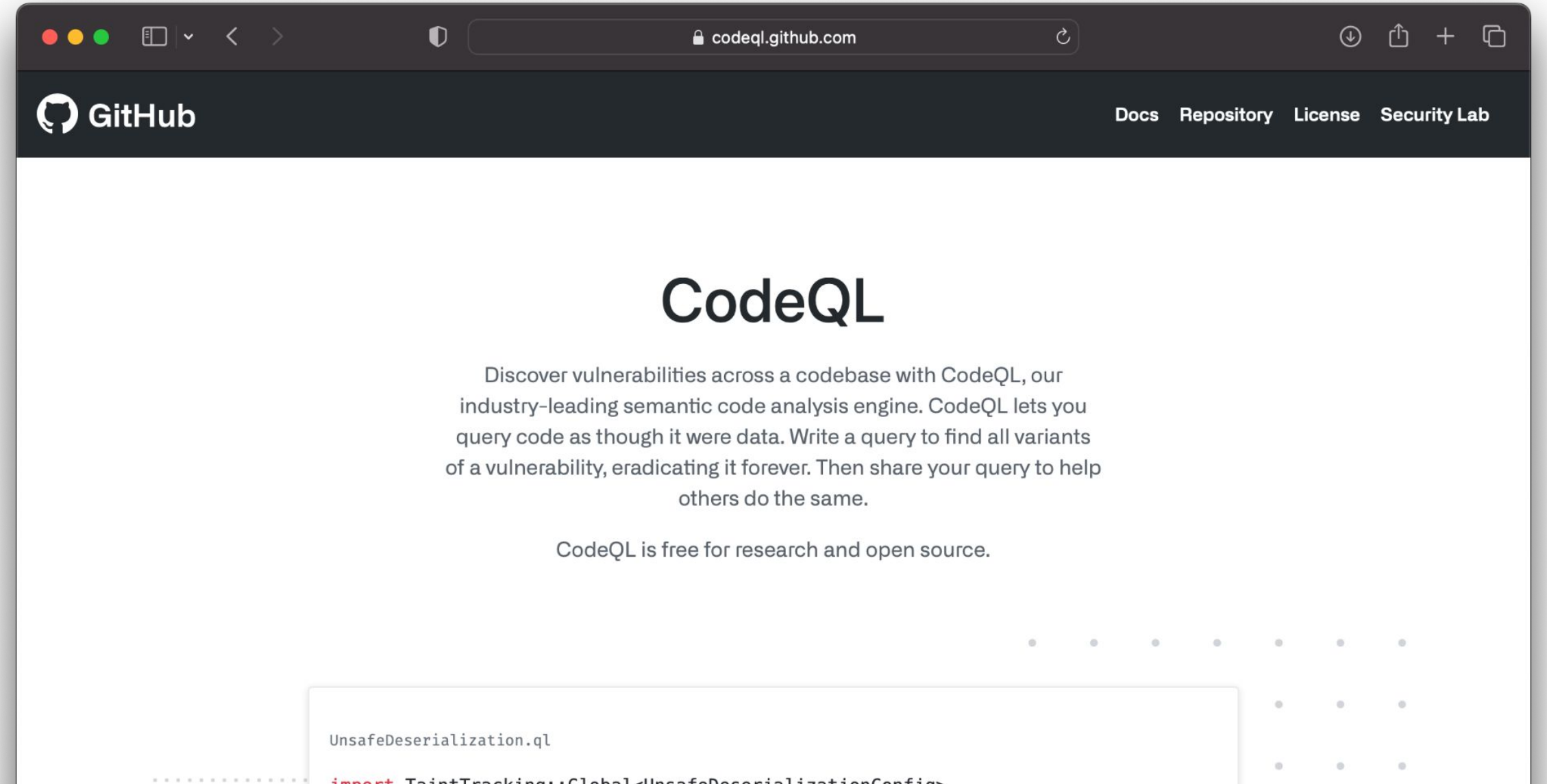
Integration Tests



CodeQL Scans

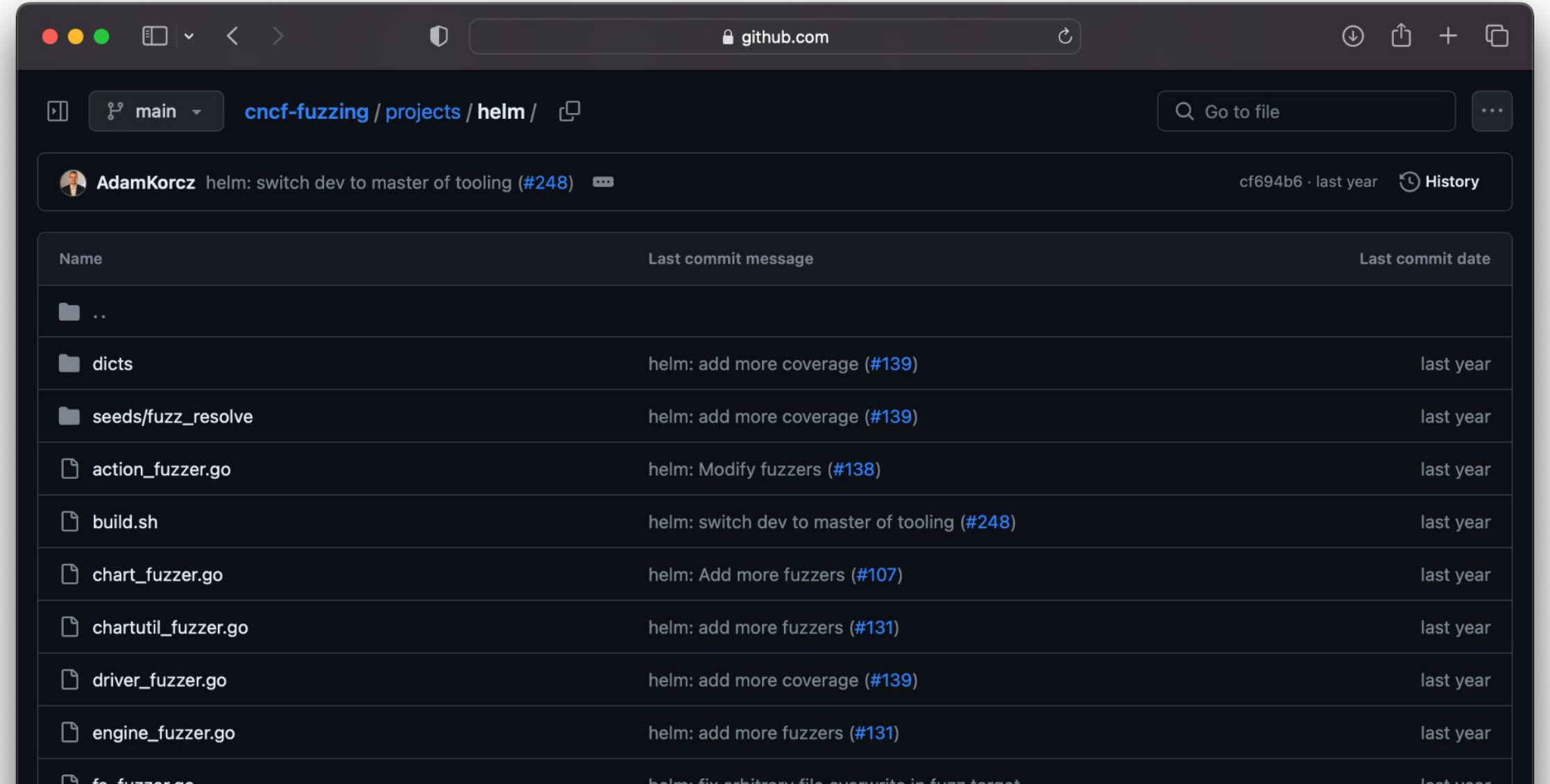


CodeQL

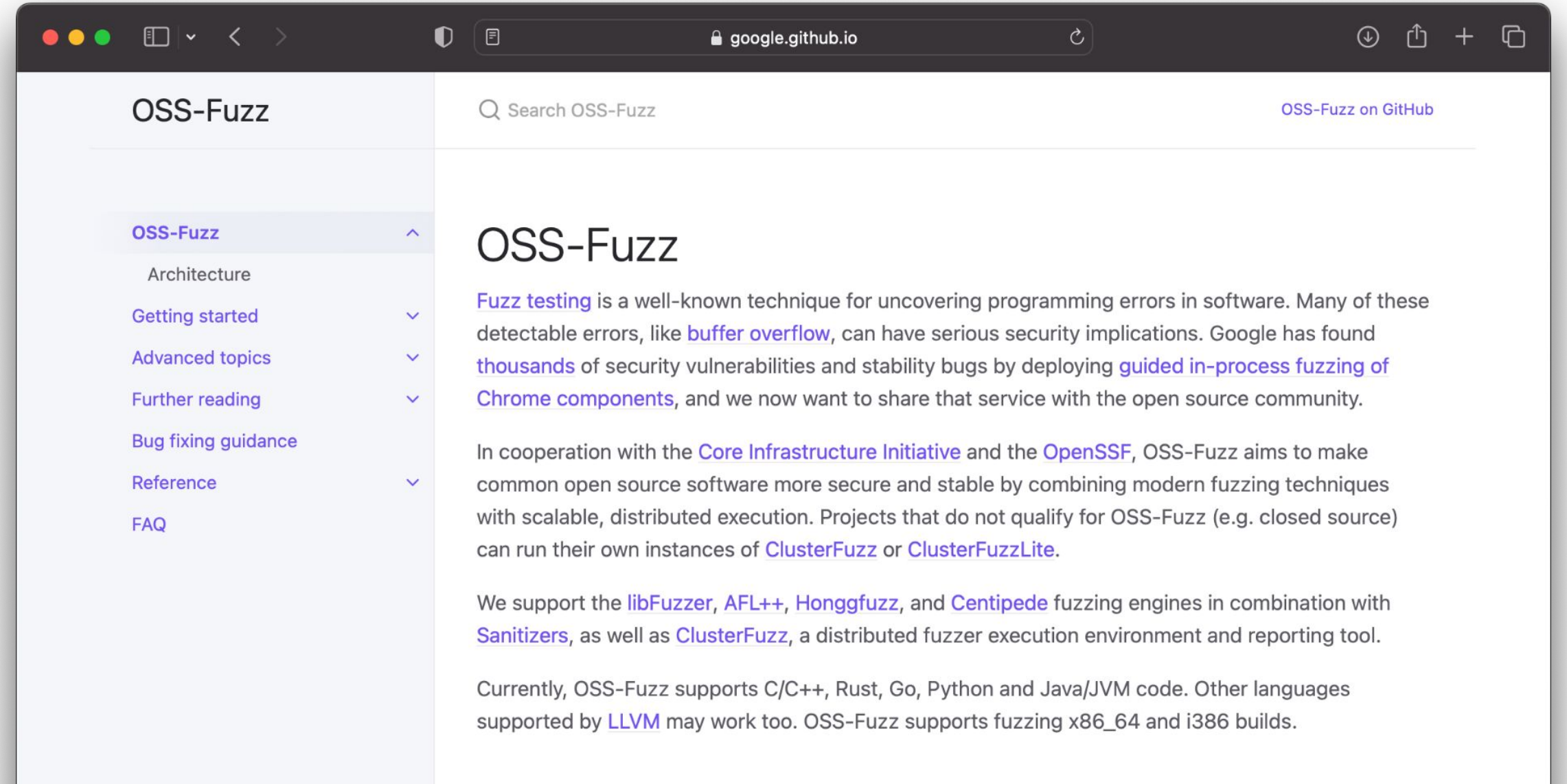

codeql.github.com
GitHub
Docs
Repository
License
Security Lab
CodeQL
Discover vulnerabilities across a codebase with CodeQL, our industry-leading semantic code analysis engine. CodeQL lets you query code as though it were data. Write a query to find all variants of a vulnerability, eradicating it forever. Then share your query to help others do the same.
CodeQL is free for research and open source.
UnsafeDeserialization.ql

Fuzz Testing

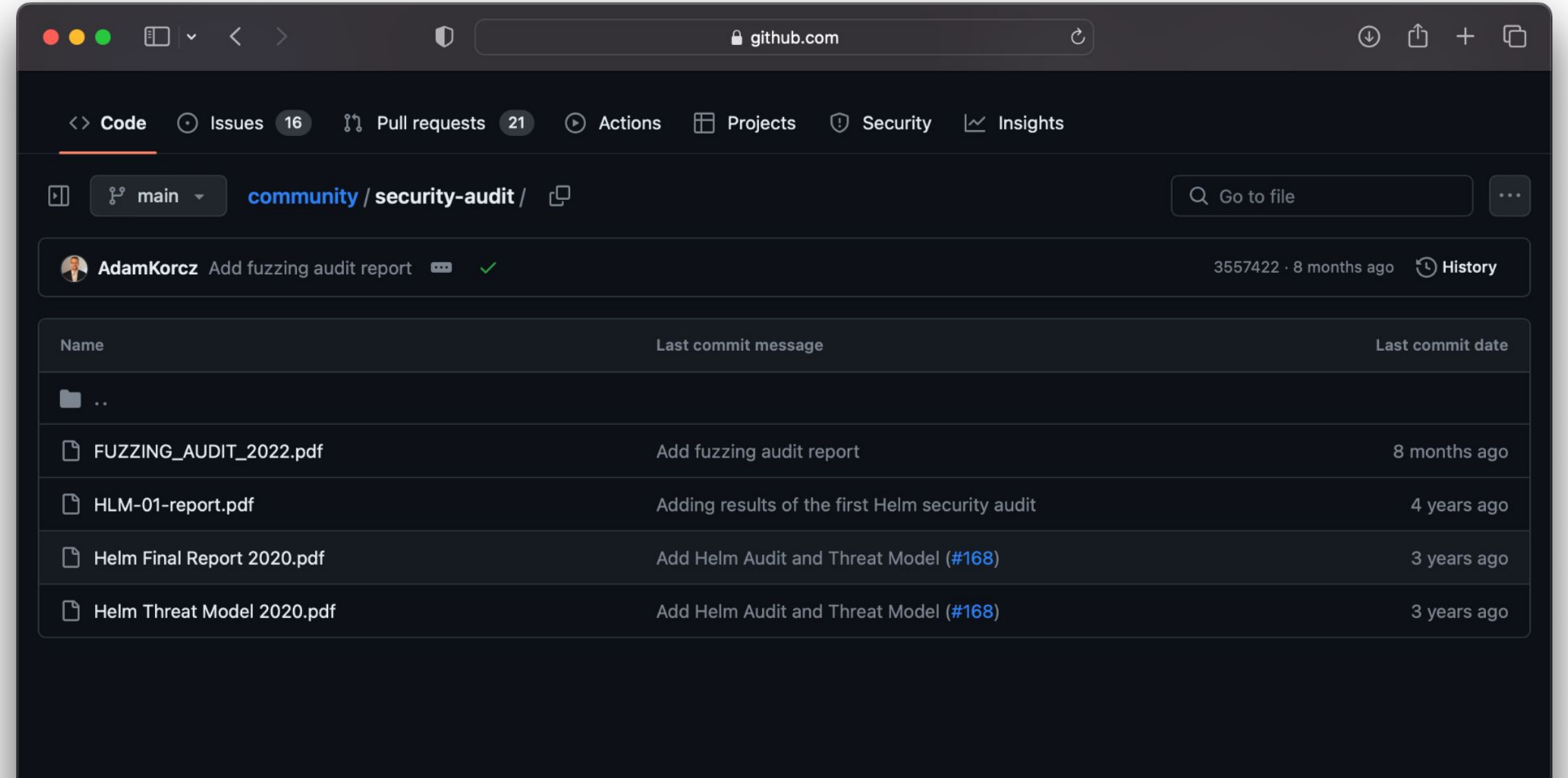
KubeCon CloudNativeCon North America 2023



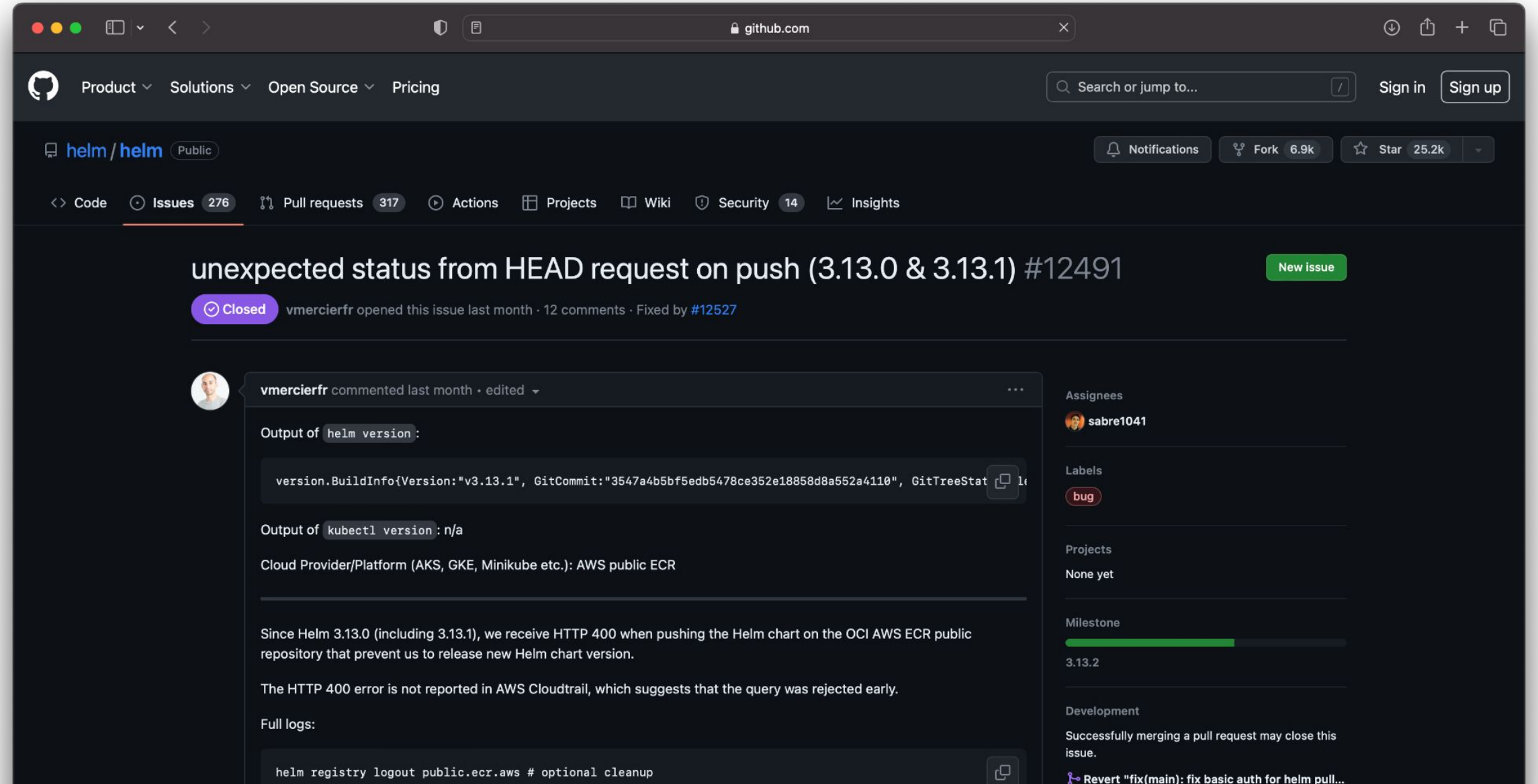
OSS-Fuzz


google.github.io
OSS-Fuzz
Search OSS-Fuzz
OSS-Fuzz on GitHub
OSS-Fuzz
Architecture
Getting started
Advanced topics
Further reading
Bug fixing guidance
Reference
FAQ
OSS-Fuzz
Fuzz testing is a well-known technique for uncovering programming errors in software. Many of these detectable errors, like buffer overflow, can have serious security implications. Google has found thousands of security vulnerabilities and stability bugs by deploying guided in-process fuzzing of Chrome components, and we now want to share that service with the open source community.
In cooperation with the Core Infrastructure Initiative and the OpenSSF, OSS-Fuzz aims to make common open source software more secure and stable by combining modern fuzzing techniques with scalable, distributed execution. Projects that do not qualify for OSS-Fuzz (e.g. closed source) can run their own instances of ClusterFuzz or ClusterFuzzLite.
We support the libFuzzer, AFL++, Honggfuzz, and Centipede fuzzing engines in combination with Sanitizers, as well as ClusterFuzz, a distributed fuzzer execution environment and reporting tool.
Currently, OSS-Fuzz supports C/C++, Rust, Go, Python and Java/JVM code. Other languages supported by LLVM may work too. OSS-Fuzz supports fuzzing x86_64 and i386 builds.

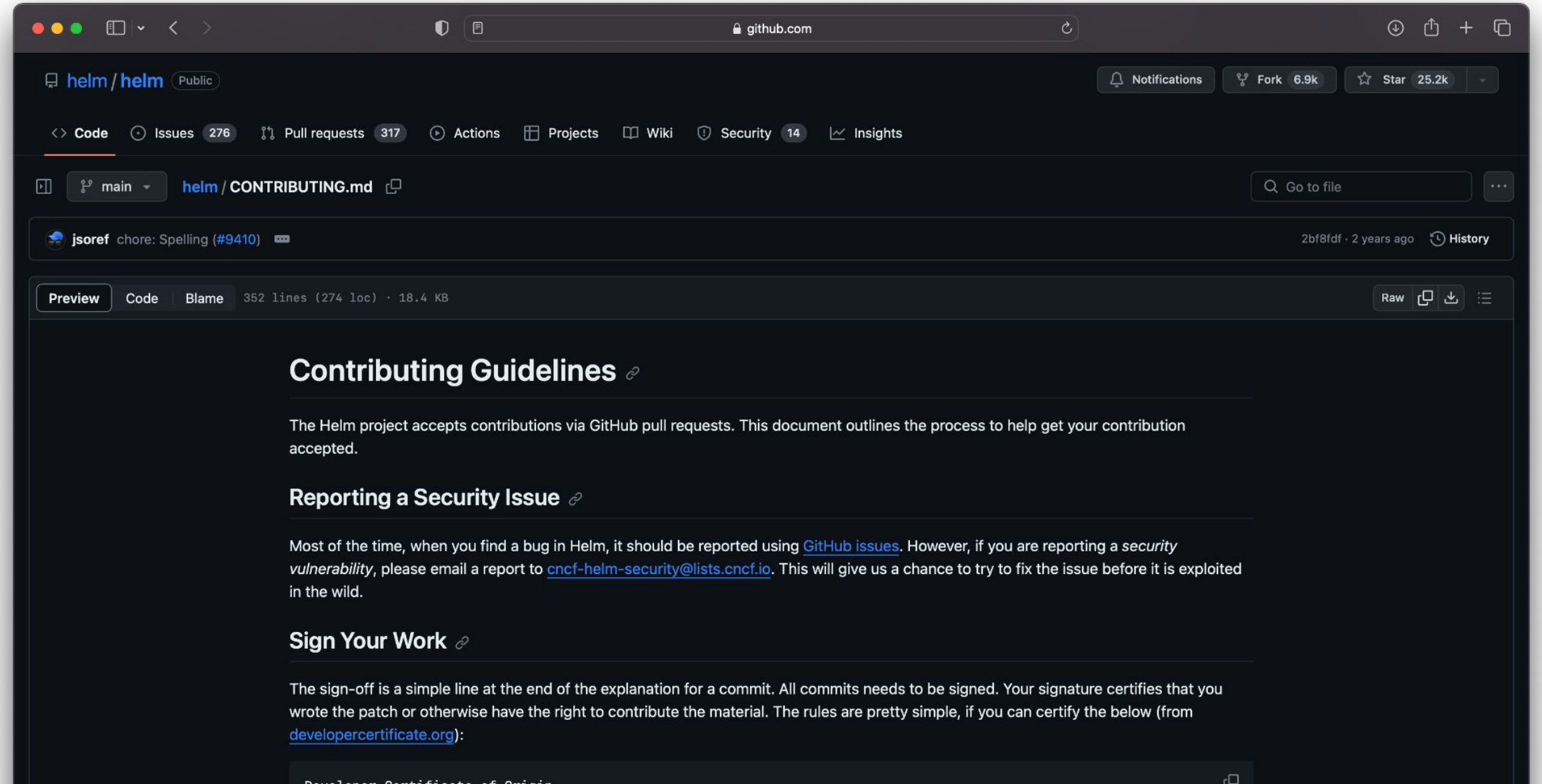
Security Audits



Oops


github.com
Product
Solutions
Open Source
Pricing
Search or jump to...
Sign in
Sign up
helm / helm
Public
Notifications
Fork 6.9k
Star 25.2k
Code
Issues 276
Pull requests 317
Actions
Projects
Wiki
Security 14
Insights
unexpected status from HEAD request on push (3.13.0 & 3.13.1) #12491
New issue
Closed
vmercierfr opened this issue last month · 12 comments · Fixed by #12527
vmercierfr commented last month • edited
Output of helm version:
version.BuildInfo{Version:"v3.13.1", GitCommit:"3547a4b5bf5edb5478ce352e18858d8a552a4110", GitTreeStat
Output of kubectl version: n/a
Cloud Provider/Platform (AKS, GKE, Minikube etc.): AWS public ECR
Since Helm 3.13.0 (including 3.13.1), we receive HTTP 400 when pushing the Helm chart on the OCI AWS ECR public repository that prevent us to release new Helm chart version.
The HTTP 400 error is not reported in AWS Cloudtrail, which suggests that the query was rejected early.
Full logs:
helm registry logout public.ecr.aws # optional cleanup
Assignees
sabre1041
Labels
bug
Projects
None yet
Milestone
3.13.2
Development
Successfully merging a pull request may close this issue.
Revert "fix(main): fix basic auth for helm pull...

Contributions



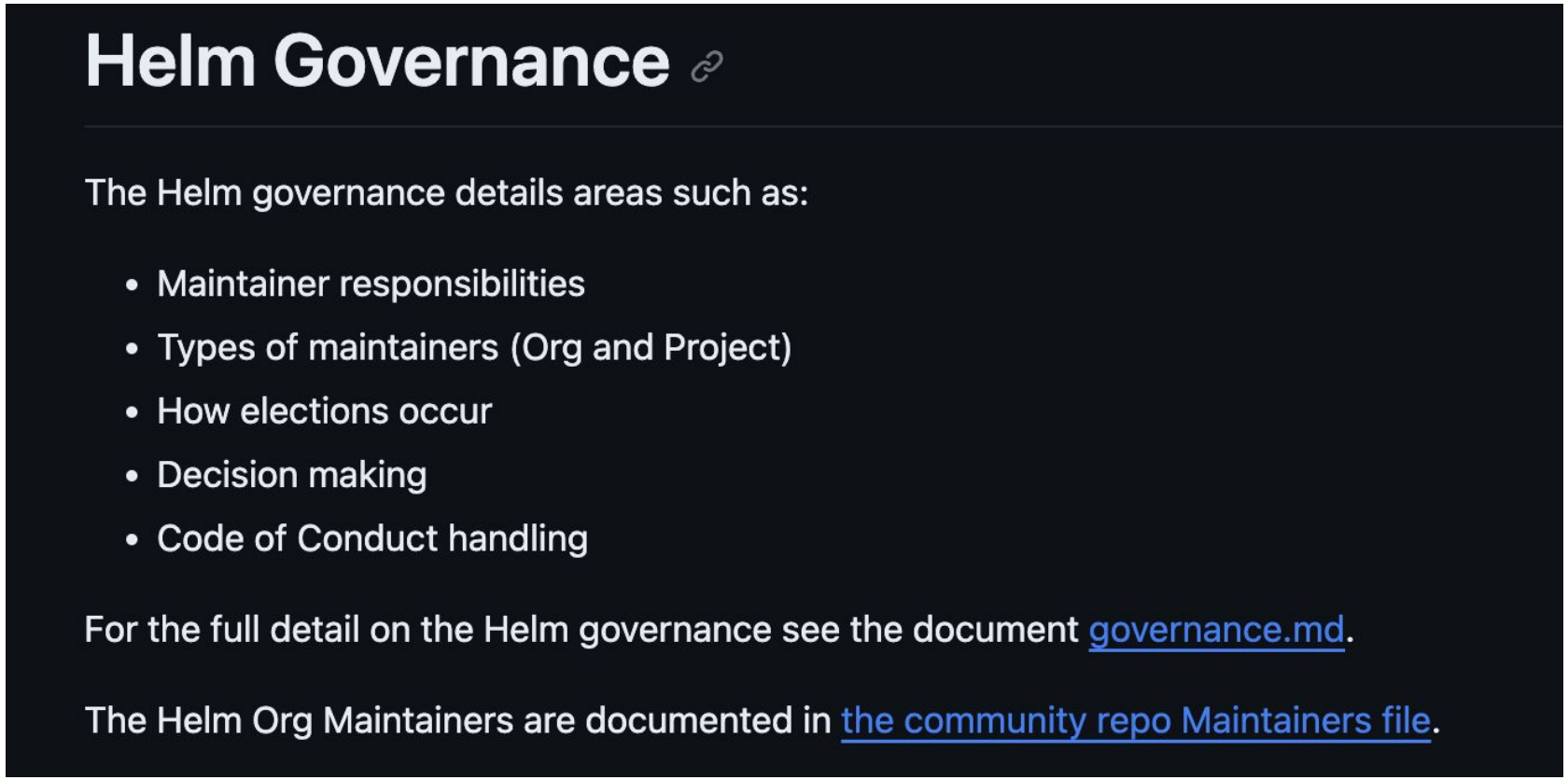

KubeCon | CloudNativeCon

North America 2023

How Helm Pulls This Off: Changing Helm (Big and Small)

Community & Governance

Strong governance model aims to ensure longevity of Helm and the Chart ecosystem



Helm Governance

The Helm governance details areas such as:

- Maintainer responsibilities
- Types of maintainers (Org and Project)
- How elections occur
- Decision making
- Code of Conduct handling

For the full detail on the Helm governance see the document governance.md.

The Helm Org Maintainers are documented in the community repo Maintainers file.

– https://github.com/helm/community/tree/main/governance

Helm Improvement Proposals

“A HIP is a design document providing information to the Helm community, or describing a new feature for a Helm project or its processes or environment.”

– https://github.com/helm/community/blob/main/hips

Quality

“Reliability is "quality changing over time""

– National Institute of Standards and Technology (https://www.itl.nist.gov/div898/handbook/apr/section1/apr111.htm)

Managing Contributions

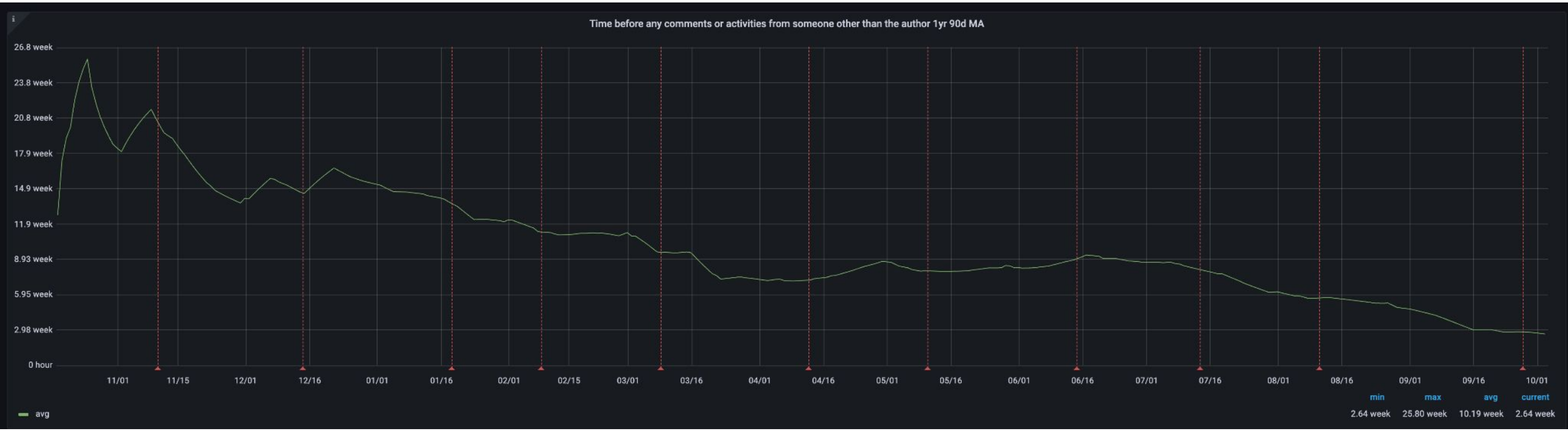
Issues: 1Yr closed: ~900

PRs:

- Prior 12 months merged: ~150
- 300 outstanding 😢
- Lots more stats: https://helm.devstats.cncf.io/

PR Reviews

We are listening and responding



Maintainers & Community

Maintainers: Triage and PR reviewers needed

Retro

Retro on Helm development and features:

- What's going well
- What are you thinking about
- What could be better

We will discuss most upvoted items.

Date: November 30 in place of the weekly developer meeting:

- 9:30-10:30 Pacific | 10-30-11:30 Mountain
- 11:30-12:30 Central | 12:30-13:30 Eastern
- 17:30-18:30 UTC

Please scan the QR Code above
to leave feedback on this session

PromCon
North America 2021

Q & A

Release Cadence

- Minor: 3 minor releases per-year (Align with Kubernetes)
- Patch: One a month, second Wednesday

User Base

Personas:

Cluster Admin

Application Manager

Chart Developer

https://github.com/helm/community/blob/main/user-profiles.md

Survey

Who uses Helm for production software delivery? Or plans to?

(show of hands)

Why/What Reliable and Usable?

What is reliable? What is usable? And to whom?

Reliable: consistently good in quality or performance; able to be trusted.

Usable: able or fit to be used.

Good/Stable/Usable

- Reliability, stability
- Where does Helm fit within the ecosystem
- Helm has staying power: designed for longevity
    - e.g. testing

Usability:

- new vs power users
-

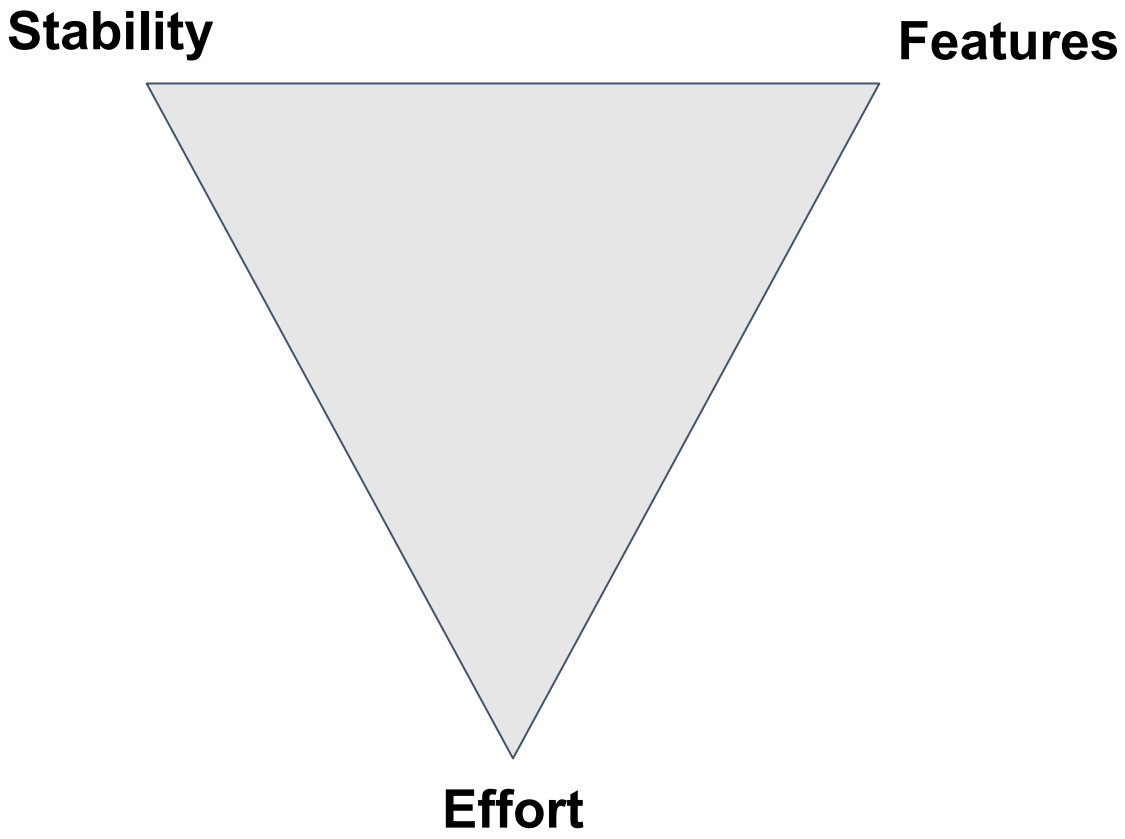
Stability

A major goal of the Helm project is to keep Helm reliable and usable.

Helm has many, many users out there. Who have incorporated Helm into a vast number of workflows and usage patterns.

In practice, Helm is used by so many people, in so many scenarios. Even small incompatibilities break users, which will erode the trust of users.

Stability vs Features



Notable Features 2023

- OCI support
- helm template –dry-run=server
- ?

Stats

Issues: 1Yr closed: ~900

PRs:

- 1Yr merged: ~150
- 300 outstanding 😢

Helm Downloads: X downloads/week

- 3.2.1: 91k downloads/last week / ~1 TB transfer
- 3.13.1:

Stars: 25k (KubeCon EU: 24.1k)

Forks: 6.9k (KubeCon EU: 6.6k)

~24,000 members in #helm-users

11,281 Helm charts on Artifact Hub (more in OCI repos)

Helm is a mature product at this point. There is reasonable inertia when it comes to making changes.

But those changes can help many users.

-