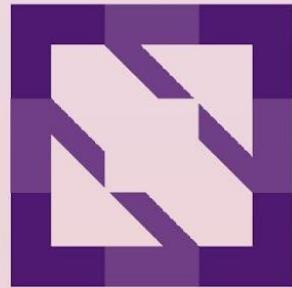




KubeCon

North America 2023



CloudNativeCon



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



KubeCon



CloudNativeCon

North America 2023

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



KubeCon



CloudNativeCon

North America 2023

~2.2 M

Latest Version Downloads

> 11 M

Total Downloads

> 151 TB

Bandwidth Transferred

Highlights: Old Version DLs Last Month

> 440k

3.9.0

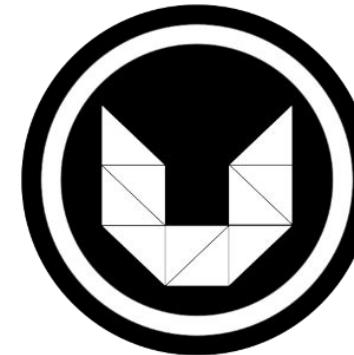
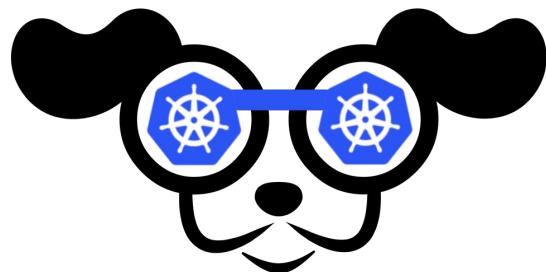
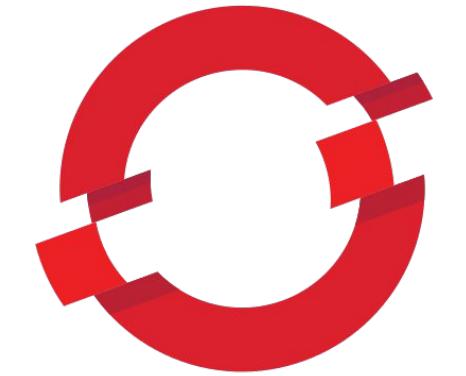
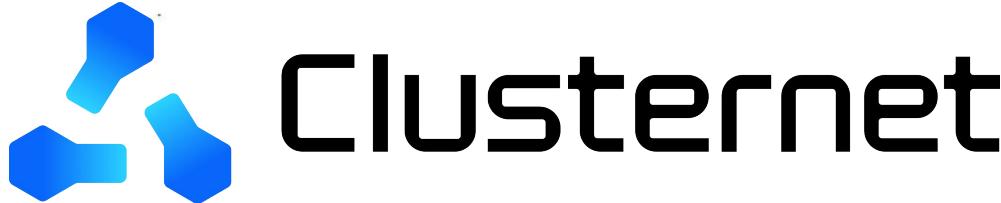
> 470k

3.2.1

> 350k

2.17.0

Applications Using Helm SDK



KubeLinter

Scripts Using Helm CLI

10s of thousands of
shell scripts on GitHub

Public Charts

The screenshot shows the ArtifactHub.io website interface. The top navigation bar includes the logo, search bar, and links for DOCS, STATS, SIGN UP, SIGN IN, and settings. A sidebar on the left contains filters for Official, Verified publishers, and CNCF, and a KIND filter section where 'Helm charts' is selected. The main content area displays three Helm chart repositories:

- kube-prometheus-stack** by Prometheus (prometheus-community) - Version 52.1.0, updated 6 days ago. Description: kube-prometheus-stack collects Kubernetes manifests, Grafana dashboards, and Prometheus rules combined with documentation and scripts. Tags: Monitoring and logging.
- ingress-nginx** by Kubernetes (ingress-nginx) - Version 4.8.3, updated 7 days ago. Description: Ingress controller for Kubernetes using NGINX as a reverse proxy and load balancer. Tags: Networking.
- cert-manager** by cert-manager (cert-manager) - Version 1.13.2, updated 2 days ago. Description: cert-manager

Helm is widely used and
expected to be mature
software.



KubeCon



CloudNativeCon

North America 2023

How Helm Pulls This Off: Scope

What Is Helm?

The screenshot shows the official Helm website (helm.sh) displayed in a dark-themed browser window. The page features a topographic map background. At the top, there is a navigation bar with links to Home, Docs, Charts, Blog, and Community, along with language selection (English) and a 'Get Started' button. The central focus is the Helm logo, which consists of the word 'HELM' in a bold, sans-serif font flanked by two stylized sun-like icons. To the right of the logo, the text 'The package manager for Kubernetes' is displayed. Below this, a callout box contains the text: 'Helm is the best way to find, share, and use software built for Kubernetes.' In the bottom left corner, there is a small icon of a ship.

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

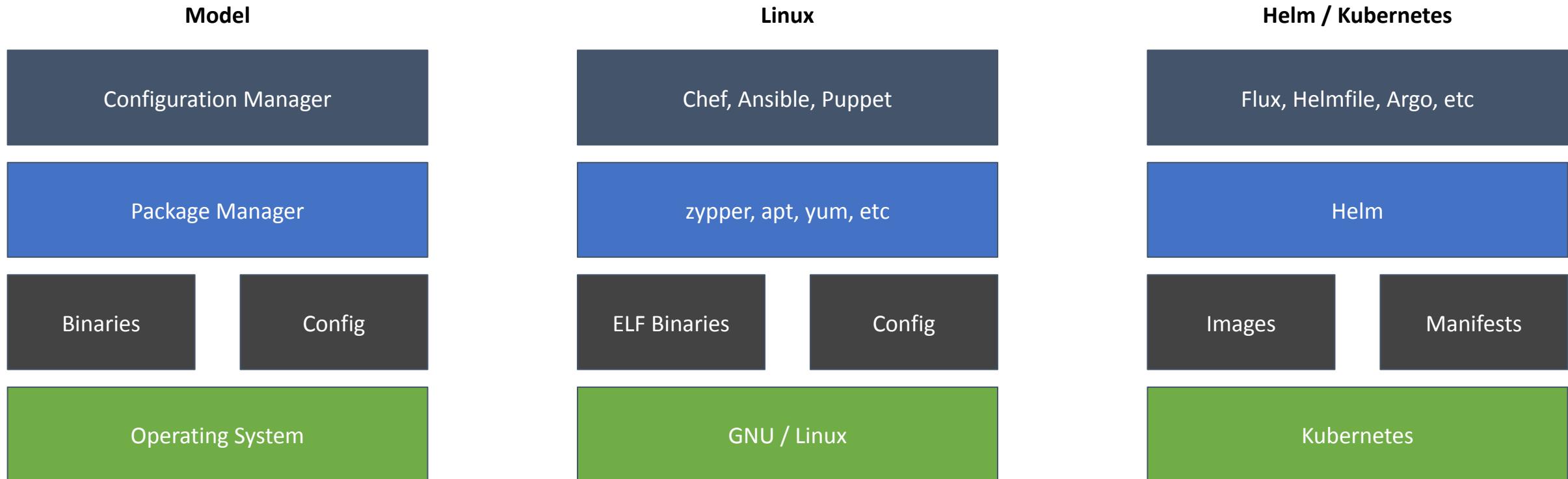
“A package manager or package-management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer in a consistent manner.”

– Wikipedia: Package Manager

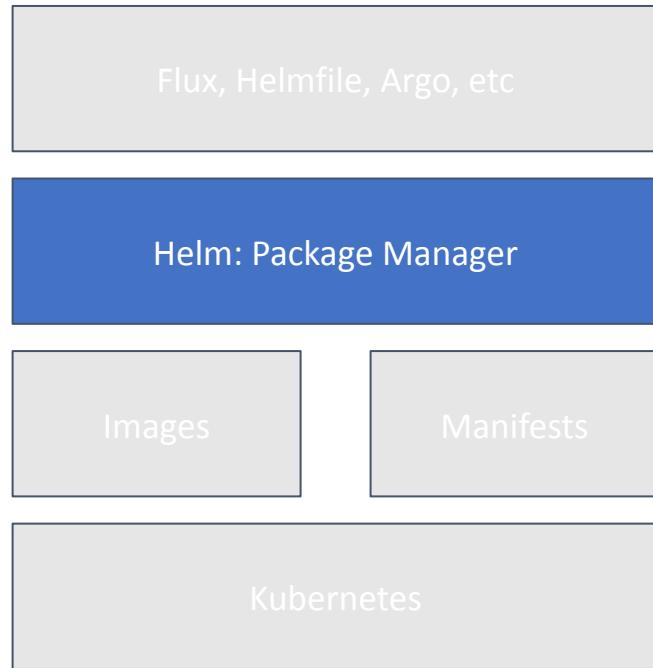
Define: Package Manager

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

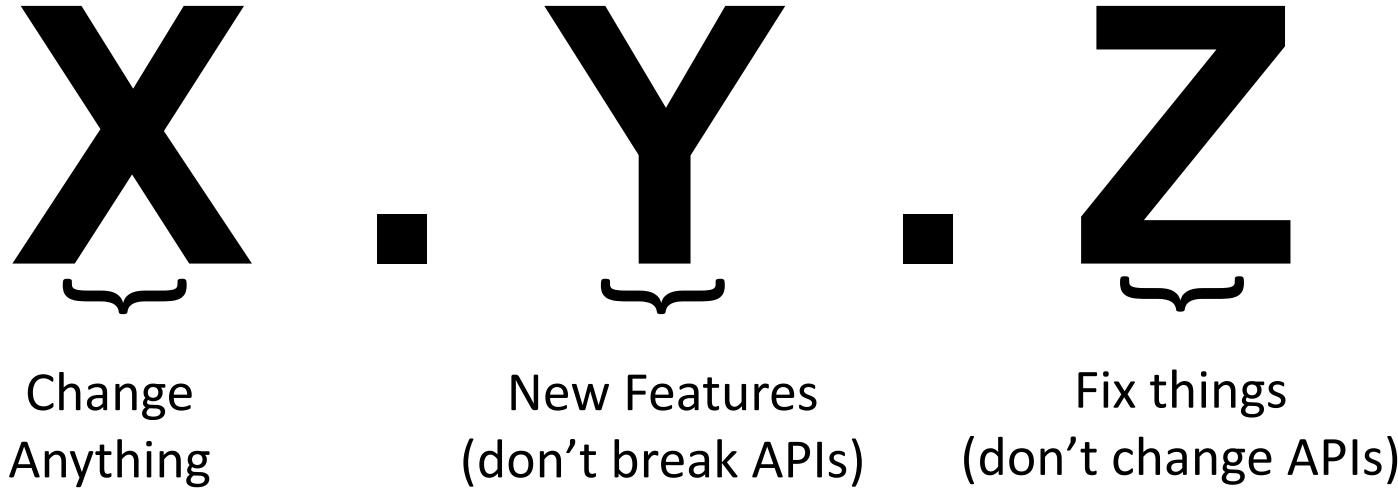


CloudNativeCon

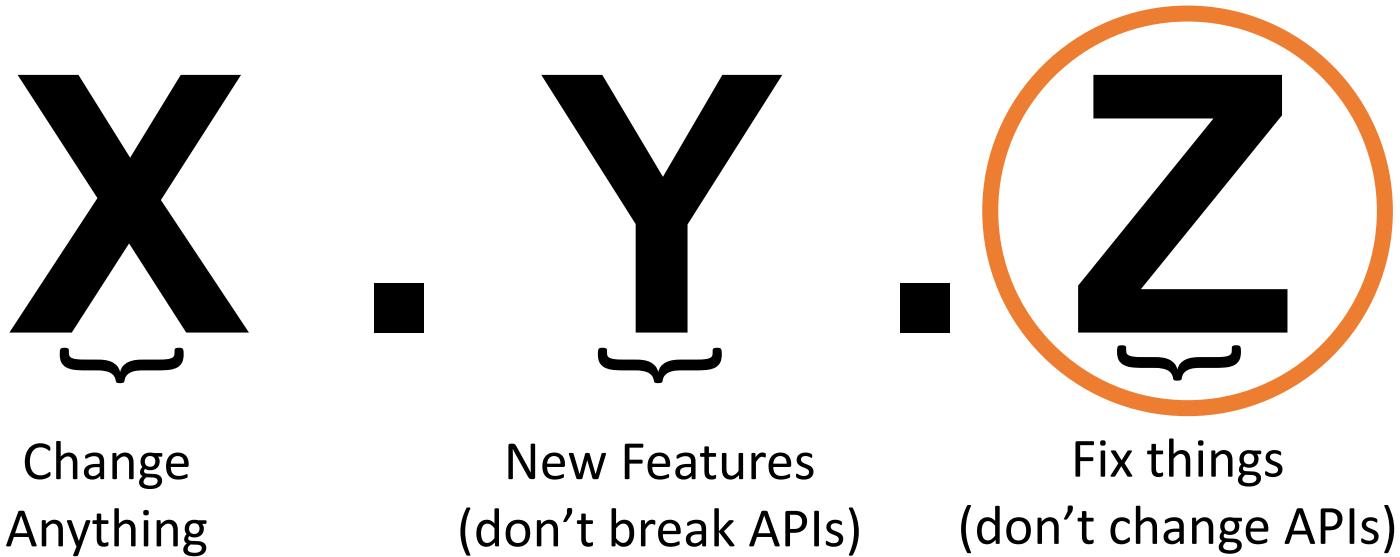
North America 2023

How Helm Pulls This Off: Versioning & Policies

Semantic Versioning



Semantic Versioning



Backwards Compatibility



KubeCon



CloudNativeCon

North America 2023

The screenshot shows a GitHub repository page for the Helm community. The repository is public and contains 16 issues, 21 pull requests, and 168 forks. The 'Code' tab is selected. A pull request titled 'community / hips / hip-0004.md' is shown, created by mattfarina last year. The pull request adds an exception to back compat. The code file 'hip-0004.md' is displayed, containing a table defining backwards-compatibility rules.

| hip | title | authors | created |
|------|--|--|------------|
| 0004 | Document backwards-compatibility rules | Marc Khouzam <marc.khouzam@montreal.ca> Matt Butcher <matt.butcher@microsoft.com> | 2020-09-18 |

Abstract

Define and document the backwards-compatibility rules that apply to Helm minor and patch releases.

Motivation

Go Module Compatibility

The screenshot shows a web browser window with the URL `go.dev` in the address bar. The page itself is a blog post titled "Keeping Your Modules Compatible" by Jean de Klerk and Jonathan Amsterdam, published on 7 July 2020. The post begins with an "Introduction" section, followed by a note about documentation on developing modules, and a warning about the challenges of releasing major version changes.

The browser interface includes a top navigation bar with links for "The Go Blog", "Why Go", "Learn", "Docs", "Packages", and "Community". Below the navigation is the main content area with the title "Keeping Your Modules Compatible" and the authors' names.

Introduction

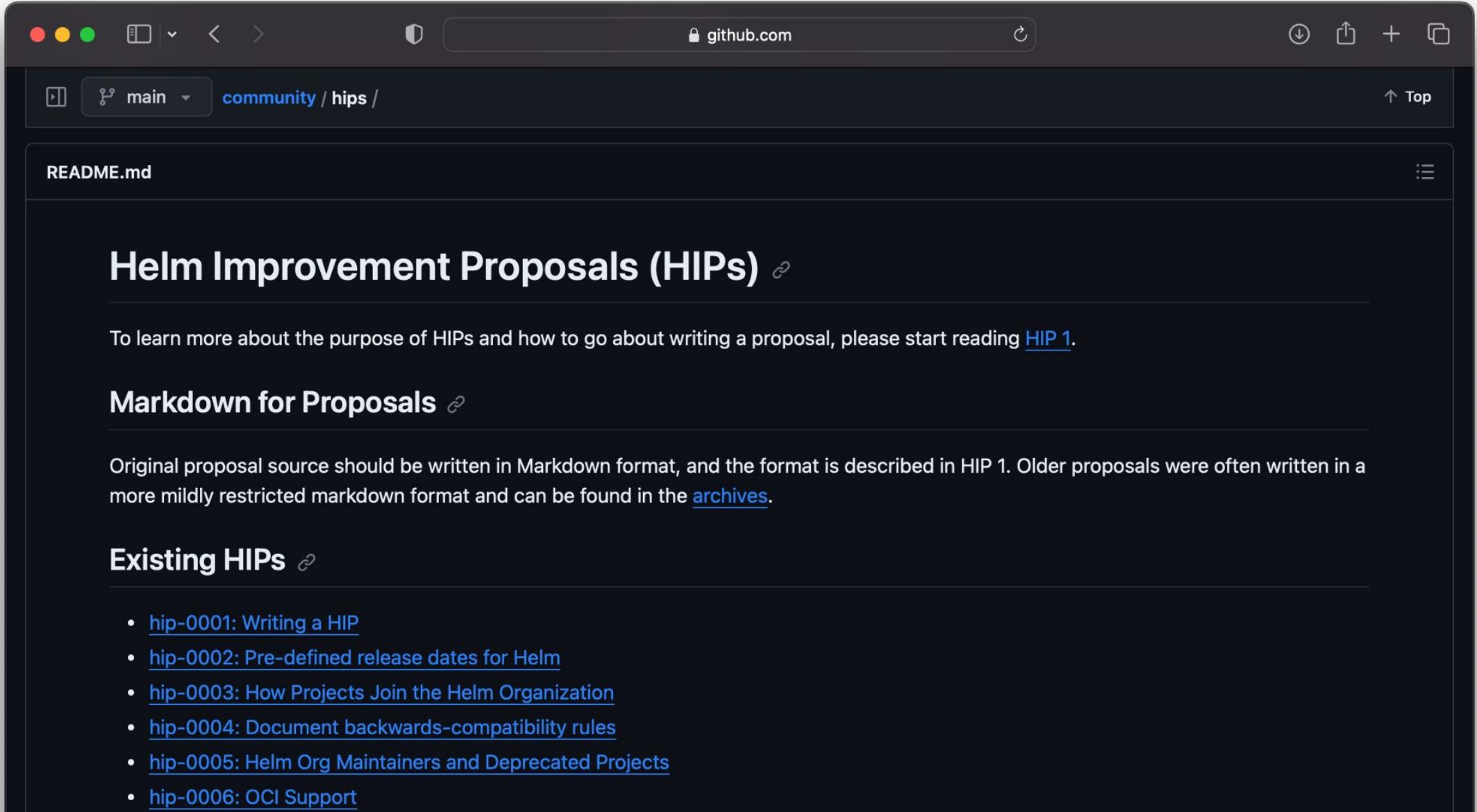
This post is part 5 in a series.

- Part 1 — [Using Go Modules](#)
- Part 2 — [Migrating To Go Modules](#)
- Part 3 — [Publishing Go Modules](#)
- Part 4 — [Go Modules: v2 and Beyond](#)
- **Part 5 — Keeping Your Modules Compatible** (this post)

Note: For documentation on developing modules, see [Developing and publishing modules](#).

Your modules will evolve over time as you add new features, change behaviors, and reconsider parts of the module's public surface. As discussed in [Go Modules: v2 and Beyond](#), breaking changes to a v1+ module must happen as part of a major version bump (or by adopting a new module path).

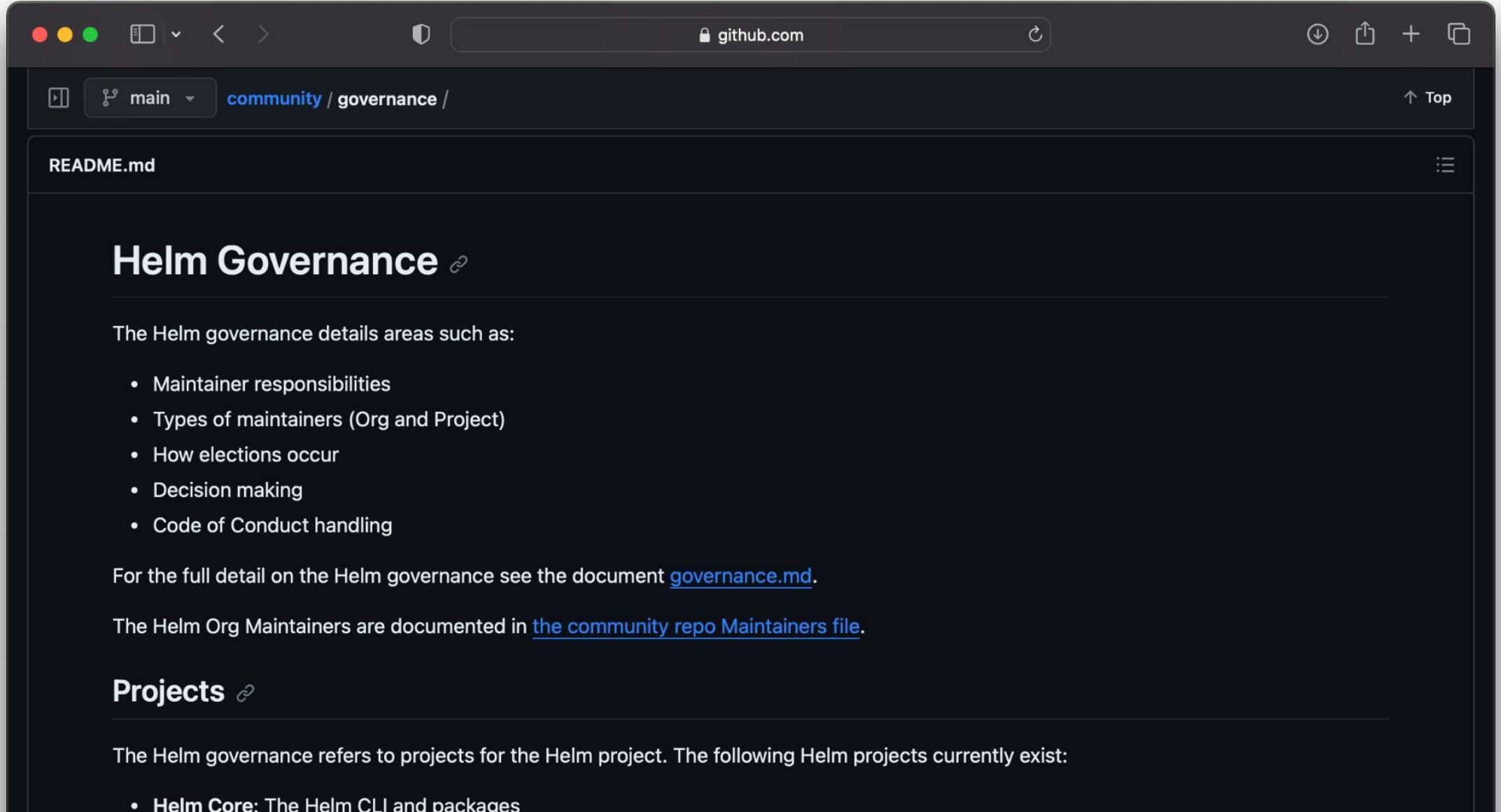
However, releasing a new major version is hard on your users. They have to find the new version, learn a new API, and change their code. And some users may never update, meaning you have to maintain two versions for your



The screenshot shows a GitHub repository page for the Helm Improvement Proposals (HIPS) at `github.com/main/community/hips/`. The page has a dark theme.

- README.md**: A section containing the file's content.
- Helm Improvement Proposals (HIPs)**: A section with a brief introduction and a link to [HIP 1](#).
- Markdown for Proposals**: A section describing the proposal source format.
- Existing HIPs**: A section listing several HIP documents:

- [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](#)



The screenshot shows a GitHub repository page for the Helm project. The URL in the address bar is `github.com/main/community/governance/`. The page title is `README.md`. The main content is titled **Helm Governance**. Below the title, it says: "The Helm governance details areas such as:" followed by a bulleted list: • Maintainer responsibilities • Types of maintainers (Org and Project) • How elections occur • Decision making • Code of Conduct handling. A note below states: "For the full detail on the Helm governance see the document [governance.md](#)." Another note says: "The Helm Org Maintainers are documented in [the community repo Maintainers file](#)." The page also has a section titled **Projects**.

The Helm governance refers to projects for the Helm project. The following Helm projects currently exist:

- **Helm Core**: The Helm CLI and packages



KubeCon

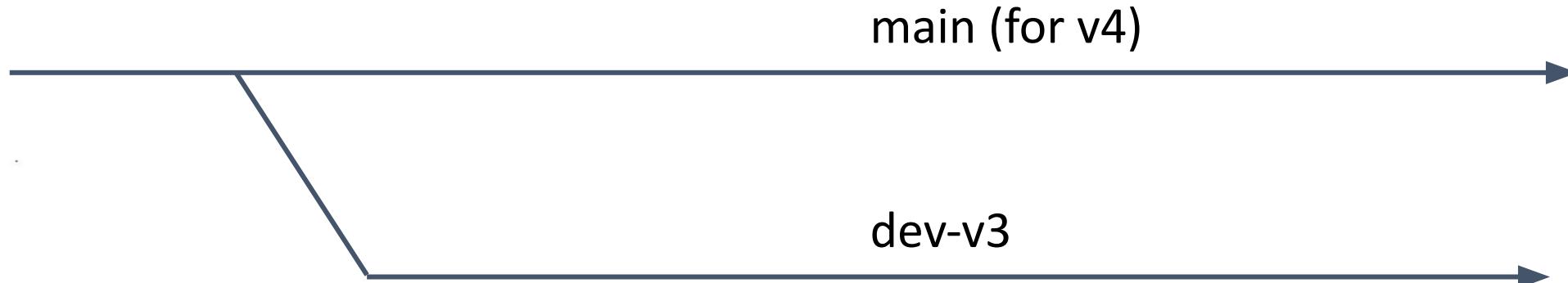


CloudNativeCon

North America 2023

Helm v4

How We Work



Helm 3 support plan

The screenshot shows a web browser window with the URL "helm.sh" in the address bar. The page header includes the Helm logo, navigation links for Home, Docs, Charts, Blog, and Community, language selection (English), and a "Get Started" button. The main content features a large title "Helm v2 Deprecation Timeline" in blue, followed by the date "Wed, Aug 12, 2020". Below the date is a quote in a box: "The time has come," the maintainers said, "To talk of software fates: Of upgrades -- and shipping Helm v3 -- Of bugfixes -- and k8s --". At the bottom, a note states: "Helm v3 was released in November 2019 , the result of ongoing community effort to evolve Helm to meet the community's needs. With a streamlined client-only experience, a renewed focus on security, and tighter".

Helm v2 Deprecation Timeline

Wed, Aug 12, 2020

with a nod to Lewis Carroll...

"The time has come," the maintainers said,
"To talk of software fates:
Of upgrades -- and shipping Helm v3 --
Of bugfixes -- and k8s --"

Helm v3 was released in November 2019 , the result of ongoing community effort to evolve Helm to meet the community's needs. With a streamlined client-only experience, a renewed focus on security, and tighter

What Do We Change?

The screenshot shows a GitHub Issues page for the `helm/helm` repository. The search bar at the top contains the query `is:open is:issue label:v4.x`. The results list several issues:

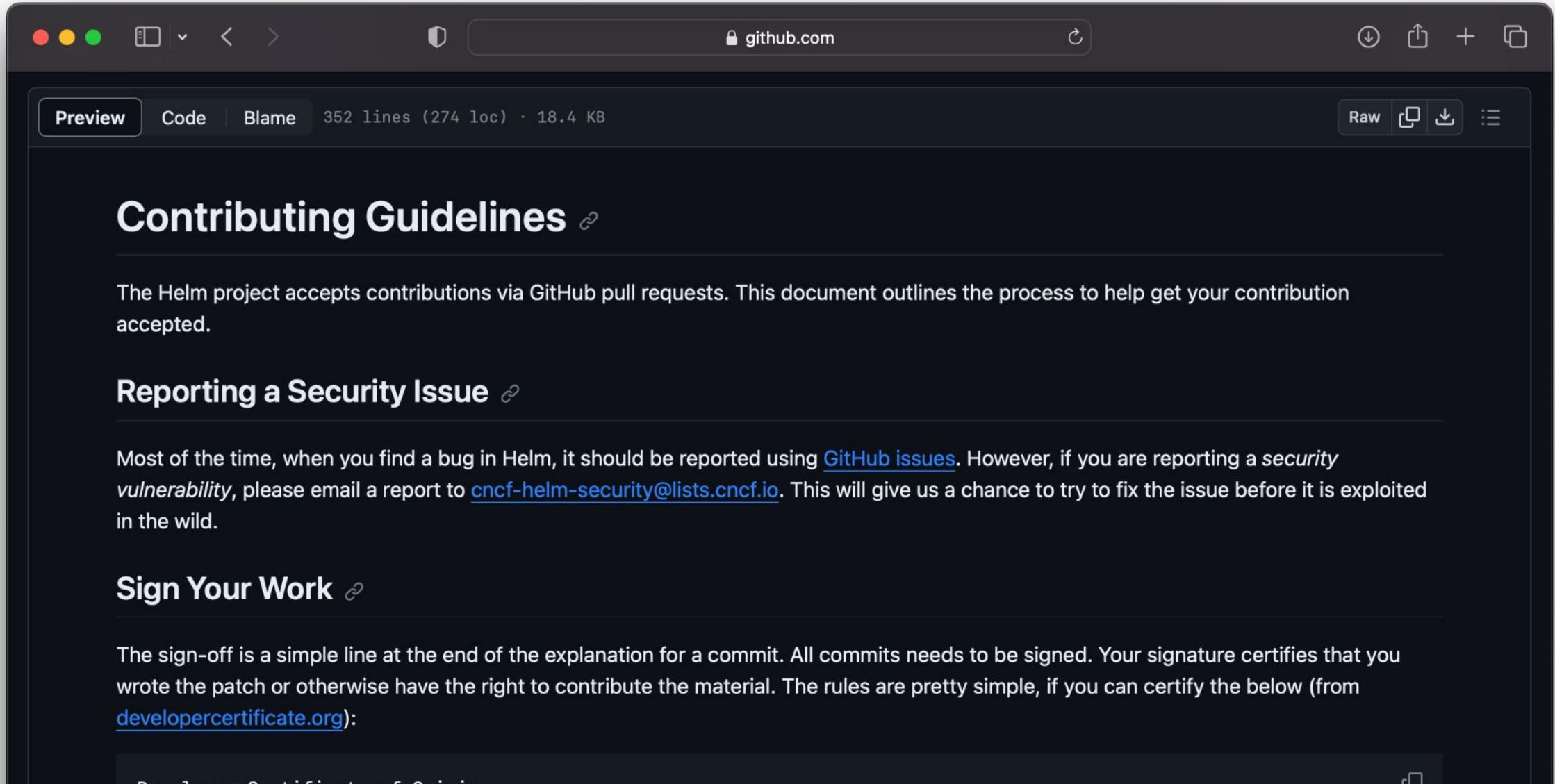
- #11493**: release secret size optimization (`feature`, `v4.x`) - opened on Nov 2, 2022 by `elops-od` - 2 comments
- #10990**: install/upgrade/template ignore unknown apiVersions in Chart.yaml (`bug`, `v4.x`) - opened on May 25, 2022 by `phil9909` - 1 comment
- #10737**: --namespace is ignored by "helm template" (`bug`, `v4.x`) - opened on Mar 8, 2022 by `jpetazzo` - 3 comments
- #10025**: Proposal: Generated charts and chart best practises (`proposal`, `v4.x`) - opened on Aug 13, 2021 by `hickeyma` - 1 comment
- #10016**: helm repo update returns 0 error code even when it fails (`proposal`, `v4.x`) - opened on Aug 10, 2021 by `knurl` - 12 comments

Not Breaking Charts

Chart.yaml:

```
apiVersion: v2
...
...
```

Want To Help?



The screenshot shows a GitHub pull request page for the Helm project. The title of the pull request is "Contributing Guidelines". The page content includes sections for reporting security issues and signing work. The GitHub interface at the top shows tabs for Preview, Code, and Blame, and various download and copy options.

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 need 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

A screenshot of a GitHub Actions build log for a pull request. The URL in the browser is <https://github.com/helm/helm/pull/911/actions>. The page shows the Actions tab selected. A green checkmark icon indicates the build was successful. The title of the build is "chore(deps): bump github.com/docker/docker from 24.0.6+incompatible to 24.0.7+incompatible #911". The build summary shows it succeeded 4 days ago in 6m 42s. The build steps listed are: Set up job (2s), Checkout source code (1s), Setup Go (0s), Install golangci-lint (1s), Test style (2m 39s), and Run unit tests (0s). The sidebar on the left shows other tabs like Code, Issues, Pull requests, Projects, Wiki, Security, and Insights.

← Back to pull request #12535

chore(deps): bump github.com/docker/docker from 24.0.6+incompatible to 24.0.7+incompatible #911 Sign in to view logs

Summary

Jobs

build

build
succeeded 4 days ago in 6m 42s

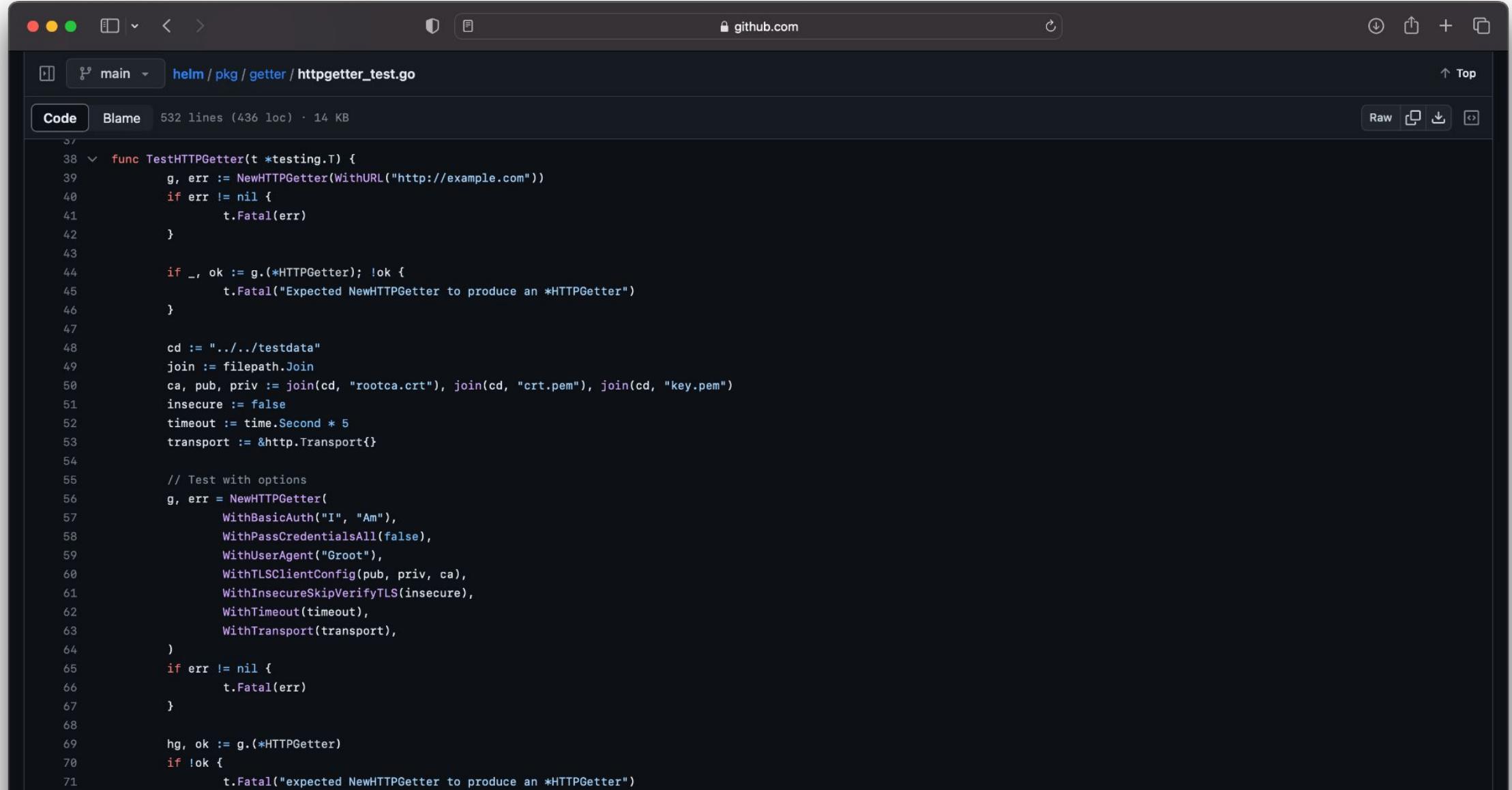
- ✓ Set up job 2s
- ✓ Checkout source code 1s
- ✓ Setup Go 0s
- ✓ Install golangci-lint 1s
- ✓ Test style 2m 39s
- ✓ Run unit tests 0s

Run details

Usage

Workflow file

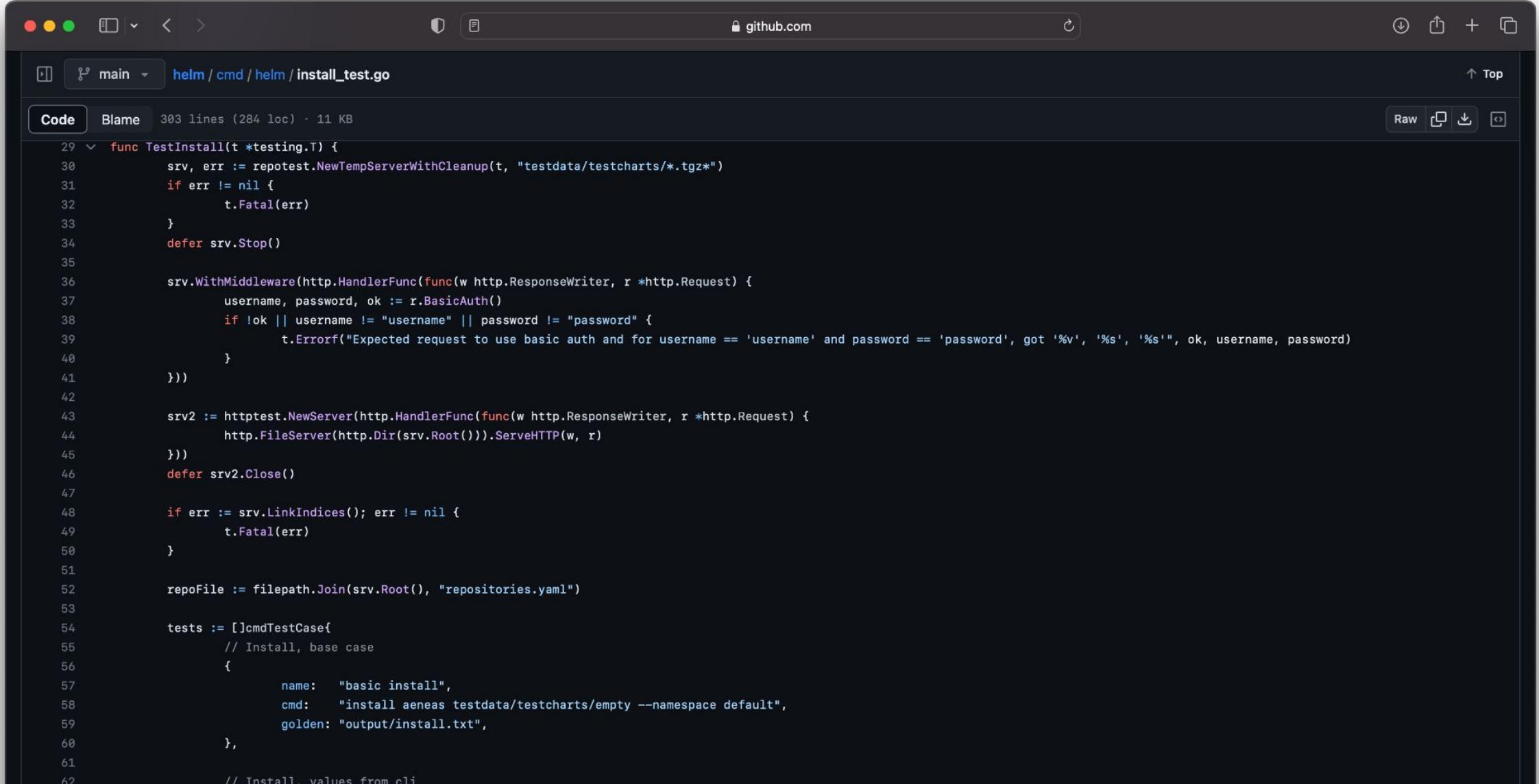
Unit and API



The screenshot shows a GitHub browser window displaying a Go test file named `httpgetter_test.go` from the `helm/pkg/getter` repository. The file contains unit tests for the `NewHTTPGetter` function. The code includes assertions for basic authentication, pass credentials, user agent, TLS client config, insecure skip verify, timeout, and transport. The GitHub interface includes standard navigation controls, a search bar, and a code editor with syntax highlighting.

```
38 func TestHTTPGetter(t *testing.T) {
39     g, err := NewHTTPGetter(WithURL("http://example.com"))
40     if err != nil {
41         t.Fatal(err)
42     }
43
44     _, ok := g.(*HTTPGetter); !ok {
45         t.Fatal("Expected NewHTTPGetter to produce an *HTTPGetter")
46     }
47
48     cd := "../../testdata"
49     join := filepath.Join
50     ca, pub, priv := join(cd, "rootca.crt"), join(cd, "crt.pem"), join(cd, "key.pem")
51     insecure := false
52     timeout := time.Second * 5
53     transport := &http.Transport{}
54
55     // Test with options
56     g, err = NewHTTPGetter(
57         WithBasicAuth("I", "Am"),
58         WithPassCredentialsAll(false),
59         WithUserAgent("Groot"),
60         WithTLSClientConfig(pub, priv, ca),
61         WithInsecureSkipVerifyTLS(insecure),
62         WithTimeout(timeout),
63         WithTransport(transport),
64     )
65     if err != nil {
66         t.Fatal(err)
67     }
68
69     hg, ok := g.(*HTTPGetter)
70     if !ok {
71         t.Fatal("expected NewHTTPGetter to produce an *HTTPGetter")
```

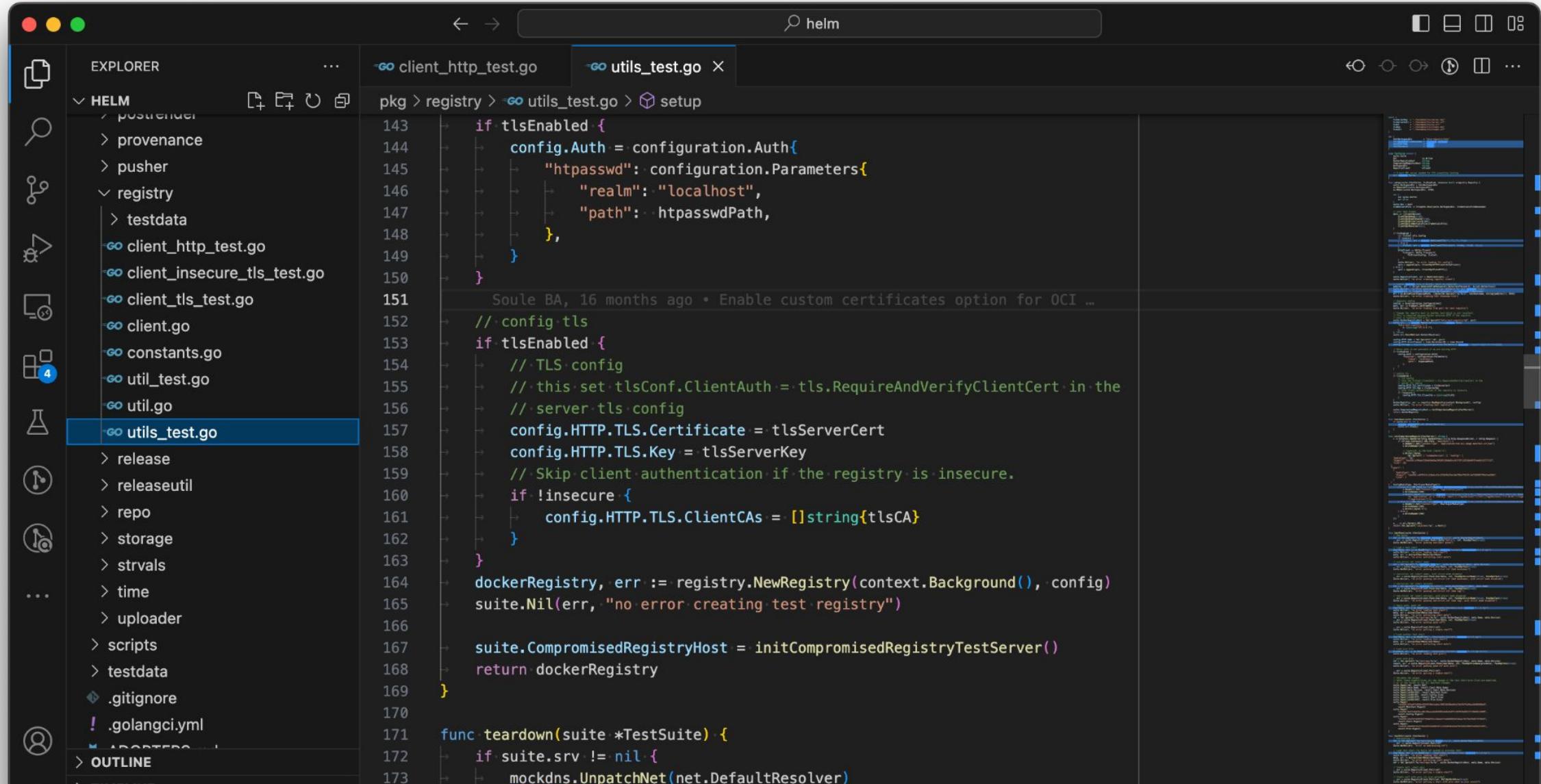
Testing CLI Output



A screenshot of a GitHub code editor interface. The URL in the address bar is `github.com`. The repository path is `helm / cmd / helm / install_test.go`. The tab bar shows "main". The "Code" tab is selected, displaying 303 lines of Go code. The code is a test function named `TestInstall` that sets up a temporary server to test basic auth and repository links.

```
29 func TestInstall(t *testing.T) {
30     srv, err := repotest.NewTempServerWithCleanup(t, "testdata/testcharts/*.tgz*")
31     if err != nil {
32         t.Fatal(err)
33     }
34     defer srv.Stop()
35
36     srv.WithMiddleware(http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
37         username, password, ok := r.BasicAuth()
38         if !ok || username != "username" || password != "password" {
39             t.Errorf("Expected request to use basic auth and for username == 'username' and password == 'password', got '%v', '%s', '%s!', ok, username, password)
40         }
41     }))
42
43     srv2 := httptest.NewServer(http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
44         http.FileServer(http.Dir(srv.Root())).ServeHTTP(w, r)
45     }))
46     defer srv2.Close()
47
48     if err := srv.LinkIndices(); err != nil {
49         t.Fatal(err)
50     }
51
52     repoFile := filepath.Join(srv.Root(), "repositories.yaml")
53
54     tests := []cmdTestCase{
55         // Install, base case
56         {
57             name:    "basic install",
58             cmd:    "install aeneas testdata/testcharts/empty --namespace default",
59             golden: "output/install.txt",
60         },
61
62         // Install, values from cli
63     },
64 }
```

Integration Tests



The screenshot shows a dark-themed code editor interface, likely VS Code, displaying Go test code. The code is part of the `utils_test.go` file in the `client` package. The code is responsible for setting up a Docker registry for testing. It includes logic for enabling TLS, configuring certificates, and creating a new registry instance. The code editor has a sidebar with a file tree showing the project structure under the `HELM` directory.

```
pkg > registry > utils_test.go > setup
143     if tlsEnabled {
144         config.Auth = configuration.Auth{
145             "htpasswd": configuration.Parameters{
146                 "realm": "localhost",
147                 "path": htpasswdPath,
148             },
149         },
150     }
151
152 // config.tls
153 if tlsEnabled {
154     // TLS config
155     // this.set.tlsConf.ClientAuth = tls.RequireAndVerifyClientCert in the
156     // server.tls.config
157     config.HTTP.TLS.Certificate = tlsServerCert
158     config.HTTP.TLS.Key = tlsServerKey
159     // Skip client authentication if the registry is insecure.
160     if !insecure {
161         config.HTTP.TLS.ClientCAs = []string{tlsCA}
162     }
163 }
164 dockerRegistry, err := registry.NewRegistry(context.Background(), config)
165 suite.Nil(err, "no error creating test registry")
166
167 suite.CompromisedRegistryHost = initCompromisedRegistryTestServer()
168 return dockerRegistry
169 }
170
171 func teardown(suite *TestSuite) {
172     if suite.srv != nil {
173         mockdns.UnpatchNet(net.DefaultResolver)
```

CodeQL Scans



KubeCon



CloudNativeCon

North America 2023

The screenshot shows a GitHub Actions job log for a pull request. The repository is `helm/helm`. The job is titled `Analyze (go)` and was successful, completed 4 days ago in 7m 13s. The log details the steps taken:

| Step | Time |
|------------------------------|--------|
| Set up job | 5s |
| Checkout repository | 1s |
| Initialize CodeQL | 9s |
| Autobuild | 5m 30s |
| Perform CodeQL Analysis | 1m 23s |
| Post Perform CodeQL Analysis | 0s |

The screenshot shows a web browser window with the URL `codeql.github.com` in the address bar. The page itself is the GitHub CodeQL homepage. It features a large title "CodeQL" in bold black font, followed by a descriptive paragraph about the tool's purpose: "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." Below this text, it says "CodeQL is free for research and open source." At the bottom of the page, there is a snippet of a CodeQL query file named "UnsafeDeserialization.ql".

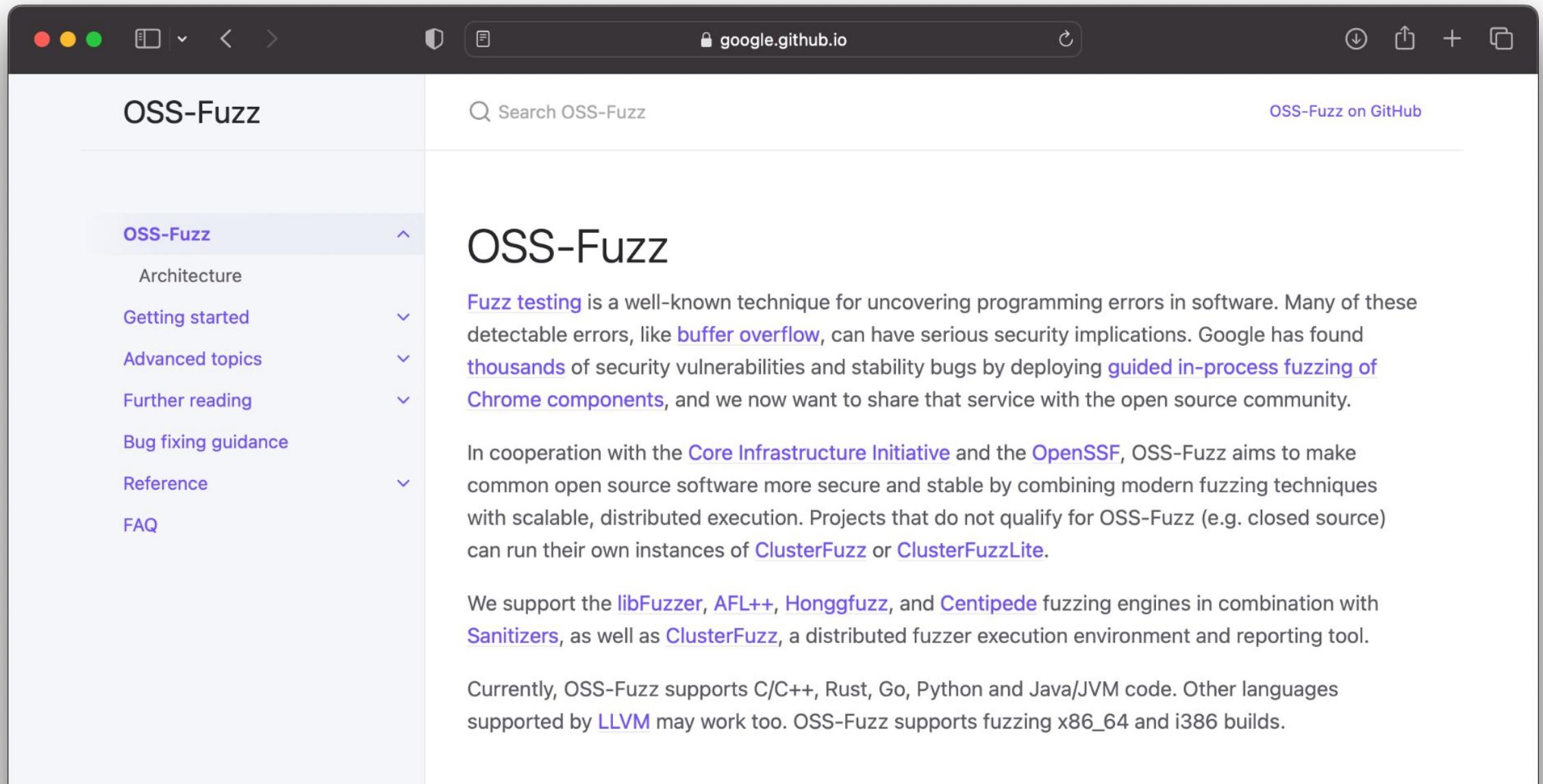
UnsafeDeserialization.ql

```
import TaintTracking::Global<UnsafeDeserializationConfig>
```

Fuzz Testing

A screenshot of a GitHub repository page for the project "helm" under the organization "cncf-fuzzing". The repository URL is "github.com/cncf-fuzzing/projects/helm". The commit history shows several commits from AdamKorcz, all related to fuzz testing. The commits are:

| Name | Last commit message | Last commit date |
|---------------------|---|------------------|
| .. | | |
| dicts | helm: add more coverage (#139) | last year |
| seeds/fuzz_resolve | helm: add more coverage (#139) | last year |
| action_fuzzer.go | helm: Modify fuzzers (#138) | last year |
| build.sh | helm: switch dev to master of tooling (#248) | last year |
| chart_fuzzer.go | helm: Add more fuzzers (#107) | last year |
| chartutil_fuzzer.go | helm: add more fuzzers (#131) | last year |
| driver_fuzzer.go | helm: add more coverage (#139) | last year |
| engine_fuzzer.go | helm: add more fuzzers (#131) | last year |
| fo_fuzzer.go | helm: fix arbitrary file overwrite in fuzz target | last year |



The screenshot shows a web browser displaying the OSS-Fuzz homepage at google.github.io. The browser interface includes standard controls (red, yellow, green dots, back, forward, search, refresh, etc.) and a tab bar.

The main content area features a large heading "OSS-Fuzz" and a search bar labeled "Search OSS-Fuzz". A link "OSS-Fuzz on GitHub" is visible in the top right. On the left, a sidebar menu titled "OSS-Fuzz" lists several sections: Architecture, Getting started, Advanced topics, Further reading, Bug fixing guidance, Reference, and FAQ. The "Getting started" section is currently expanded, showing its sub-content.

The main article discusses the purpose of OSS-Fuzz, mentioning fuzz testing, buffer overflow, thousands of vulnerabilities, and its collaboration with Core Infrastructure Initiative and OpenSSF. It also highlights support for various fuzzing engines like libFuzzer, AFL++, Honggfuzz, Centipede, and Sanitizers, as well as ClusterFuzz and ClusterFuzzLite. The article concludes by stating that OSS-Fuzz supports C/C++, Rust, Go, Python, and Java/JVM code, with LLVM-supported languages potentially working as well.

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

A screenshot of a GitHub repository page titled "community / security-audit". The repository has 16 issues and 21 pull requests. The "Code" tab is selected. The main branch is "main". A commit by AdamKorcz titled "Add fuzzing audit report" was made 8 months ago. The repository contains several PDF files related to security audits.

| Name | Last commit message | Last commit date |
|----------------------------|---|------------------|
| ... | | |
| FUZZING_AUDIT_2022.pdf | Add fuzzing audit report | 8 months ago |
| HLM-01-report.pdf | Adding results of the first Helm security audit | 4 years ago |
| Helm Final Report 2020.pdf | Add Helm Audit and Threat Model (#168) | 3 years ago |
| Helm Threat Model 2020.pdf | Add Helm Audit and Threat Model (#168) | 3 years ago |

Oops



KubeCon



CloudNativeCon

North America 2023

The screenshot shows a GitHub issue page for the Helm repository. The URL in the address bar is `github.com/helm/helm`. The repository name is `helm/helm`, labeled as Public. The main navigation tabs are `Code`, `Issues 276` (which is selected), `Pull requests 317`, `Actions`, `Projects`, `Wiki`, `Security 14`, and `Insights`. A search bar at the top right contains the placeholder `Search or jump to...`. On the far right, there are `Sign in` and `Sign up` buttons.

The issue title is `unexpected status from HEAD request on push (3.13.0 & 3.13.1) #12491`. A green button on the right says `New issue`. Below the title, it says the issue is `Closed` by `vmercierfr` last month, with 12 comments and fixed by `#12527`.

The issue body contains a comment from `vmercierfr` (last month, edited) showing the output of `helm version` and `kubectl version`. It also mentions using AWS public ECR as a Cloud Provider/Platform. The text states that since Helm 3.13.0 (including 3.13.1), they receive an HTTP 400 error when pushing the Helm chart on the OCI AWS ECR public repository, preventing them from releasing new Helm chart versions. The error is noted as not being reported in AWS CloudTrail.

On the right side of the issue page, there are sections for `Assignees` (`sabre1041`), `Labels` (`bug`), `Projects` (None yet), `Milestone` (3.13.2), and `Development` (with a note about merging pull requests). There is also a link to `Revert "fix(main): fix basic auth for helm pull..."`.

```
version.BuildInfo{Version:"v3.13.1", GitCommit:"3547a4b5bf5edb5478ce352e18858d8a552a4110", GitTreeState:"clean", GoVersion:"go1.16.6", Compiler:"gc", Platform:"linux/amd64"}
```

```
Output of helm version:
version.BuildInfo{Version:"v3.13.1", GitCommit:"3547a4b5bf5edb5478ce352e18858d8a552a4110", GitTreeState:"clean", GoVersion:"go1.16.6", Compiler:"gc", Platform:"linux/amd64"}
```

```
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
```

New issue

Closed vmercierfr opened this issue last month · 12 comments · Fixed by #12527

vmercierfr commented last month · edited

Assignees

sabre1041

Labels

bug

Projects

None yet

Milestone

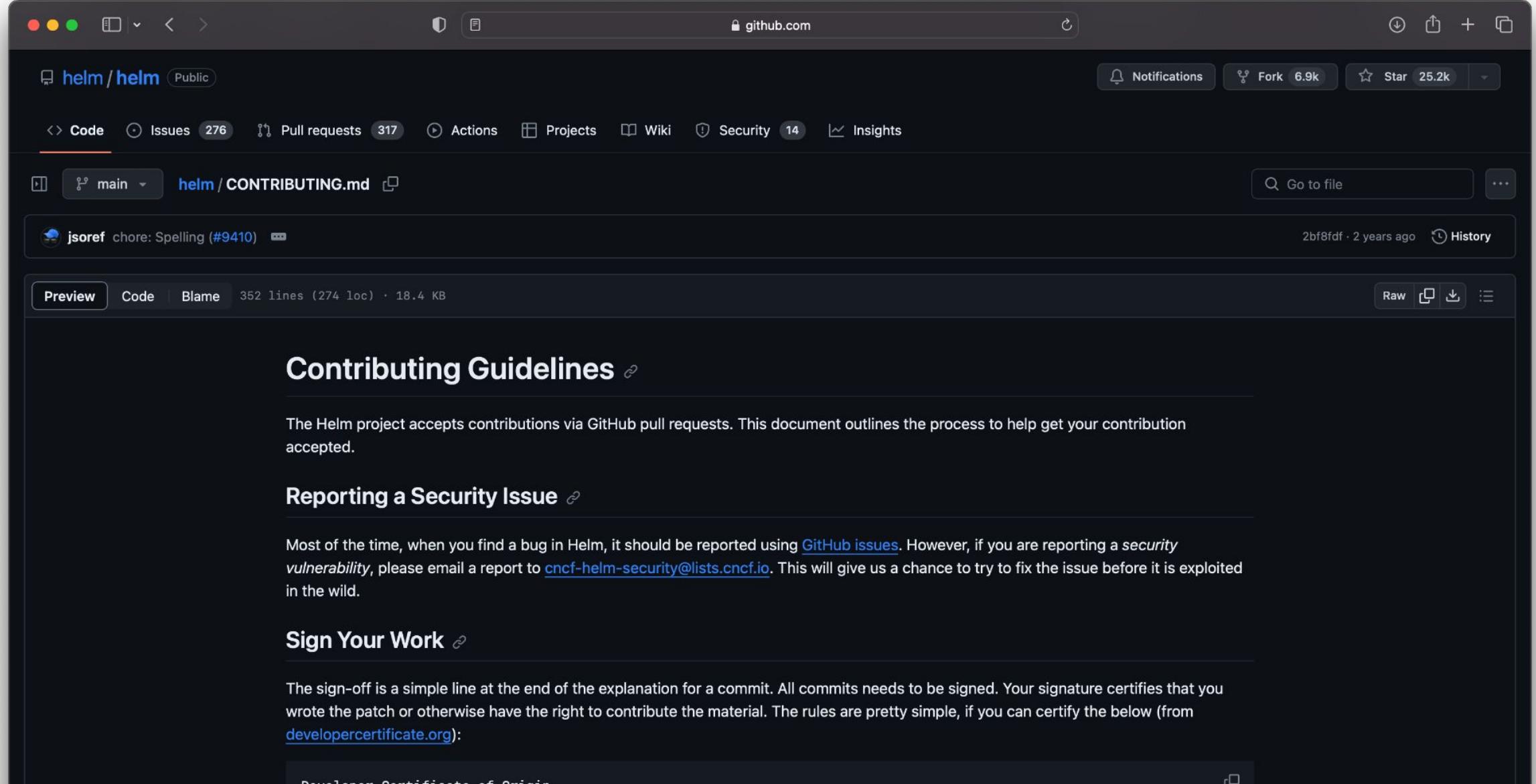
3.13.2

Development

Successfully merging a pull request may close this issue.

Revert "fix(main): fix basic auth for helm pull..."

Contributions



The screenshot shows a GitHub repository page for the Helm project. The URL in the address bar is `github.com/helm/helm`. The repository is public, has 276 issues, 317 pull requests, and 14 actions. The main branch is `main`, and the file being viewed is `helm/CONTRIBUTING.md`. A recent commit by `jsoref` titled "chore: Spelling (#9410)" was made 2 years ago. The file content is displayed in a code editor-like interface with tabs for Preview, Code, Blame, and Raw. The content includes sections on Contributing Guidelines, Reporting a Security Issue, and Sign Your Work.

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 need 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: 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>

“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



KubeCon



CloudNativeCon

North America 2023

We are listening and responding



Maintainers & Community

Maintainers: Triage and PR reviewers needed

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



PromCon
North America 2021



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



KubeCon



CloudNativeCon

North America 2023

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
-

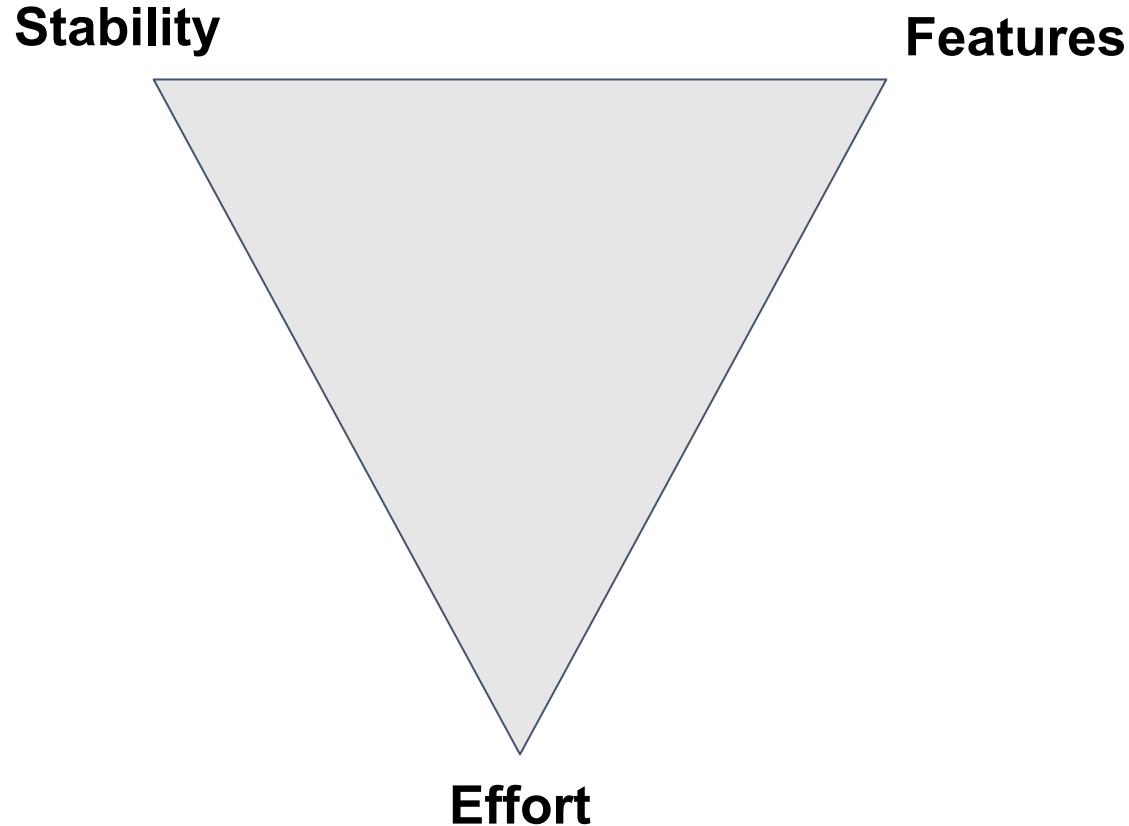
Stability

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



KubeCon



CloudNativeCon

North America 2023

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.

