KubeCon CloudNativeCon
North America 2025

Introducing Helm 4

#KubeCon #CloudNativeCon

Releasing Helm 4

KubeCon | CloudNativeCon
North America 2025

Who we are

Matt Farina

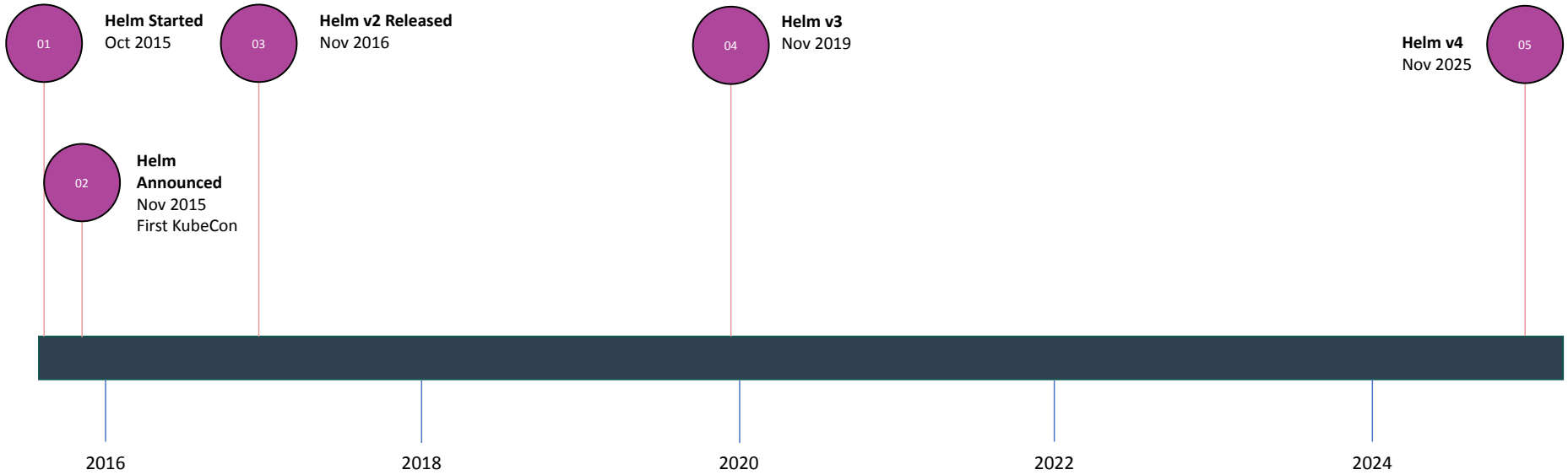
- Open Source Contributor and Maintainer
- Distinguished Engineer
- Committed Vegan

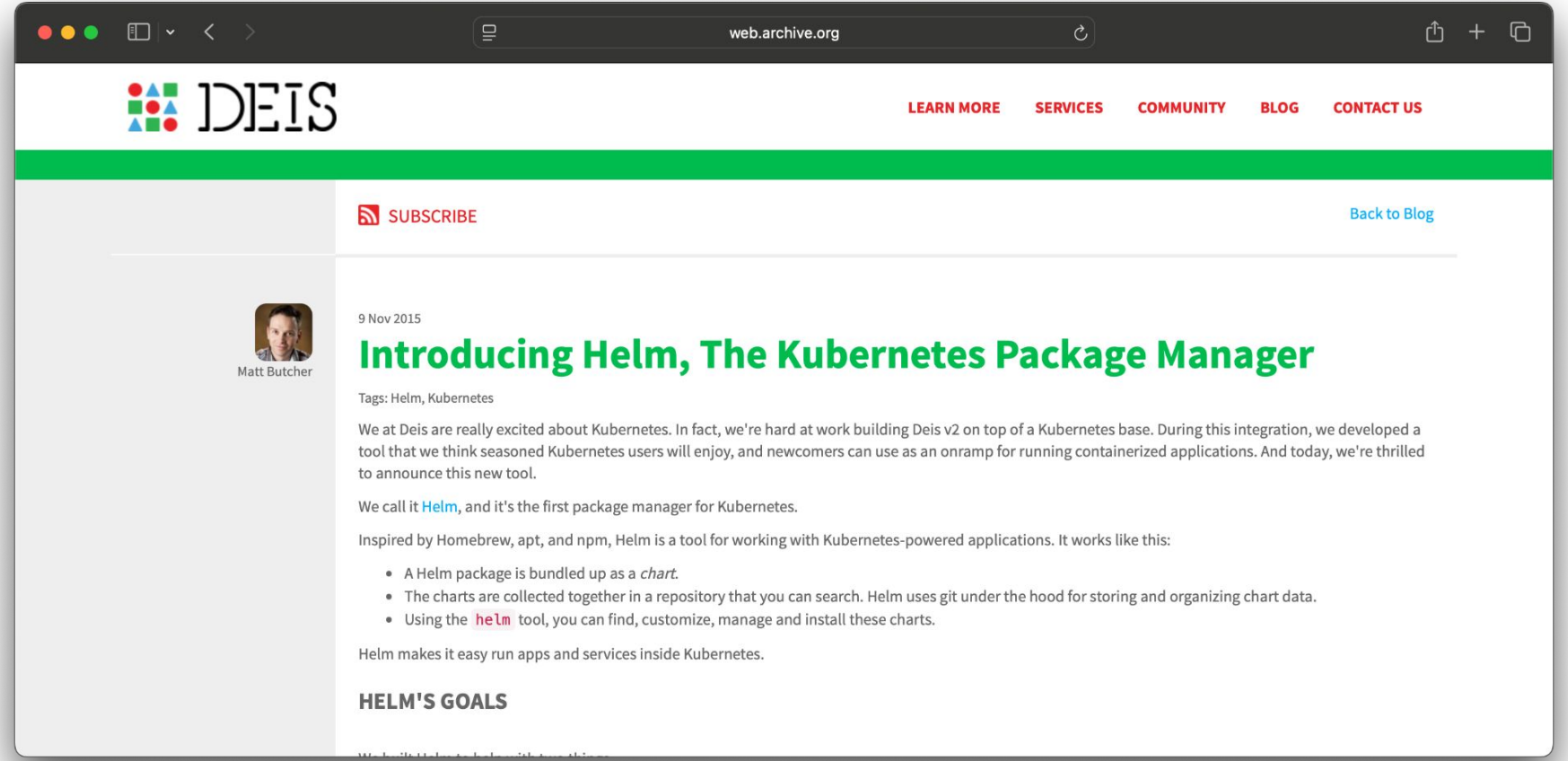
Robert Sirchia

- Open Source Contributor and Maintainer
- Director Technical & Community Marketing
- Avid Hunter

Some History

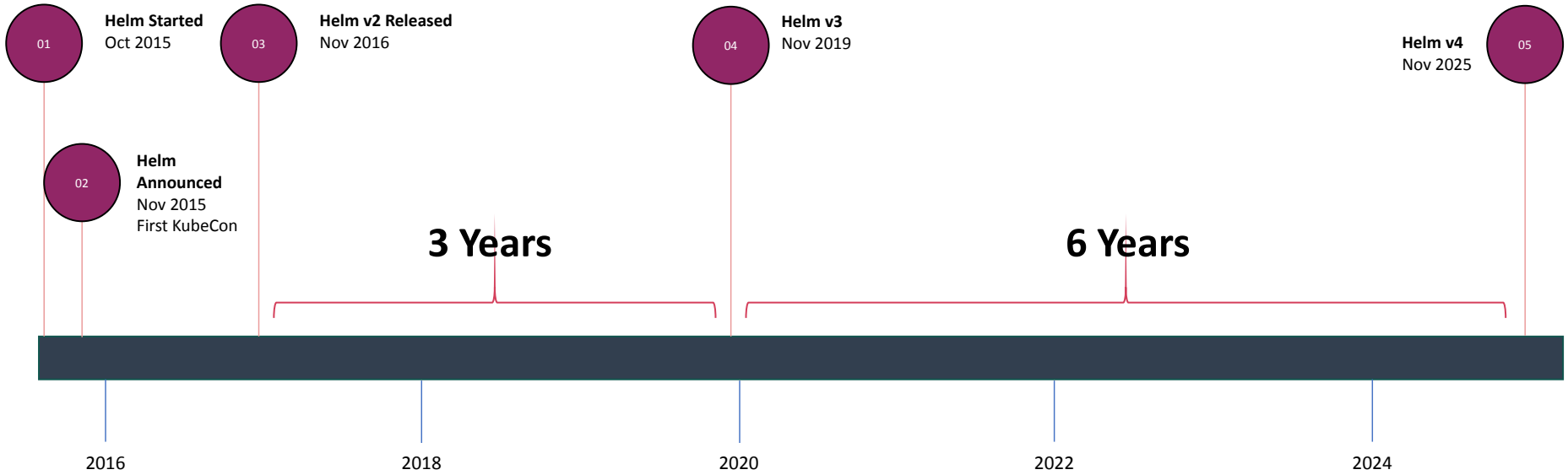
Helm At 10 Years Old



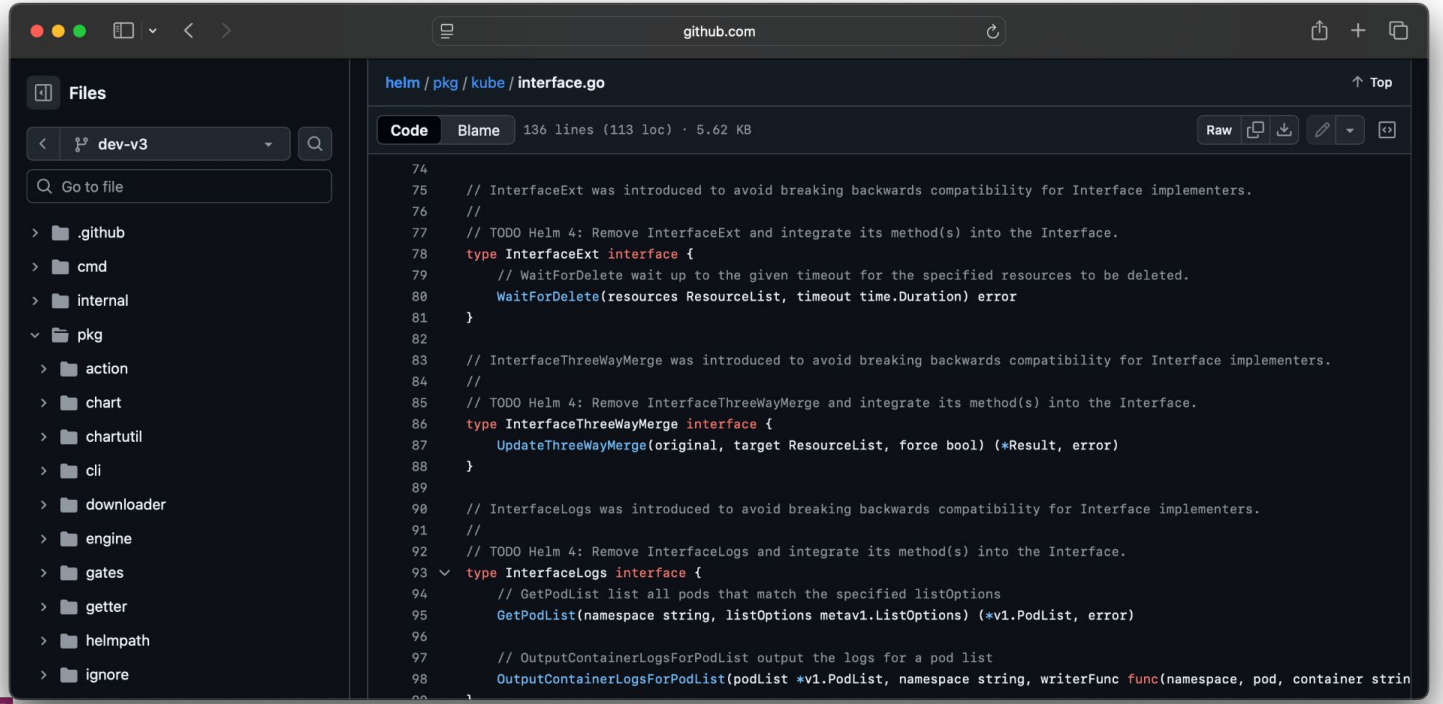

web.archive.org
DEIS
LEARN MORE
SERVICES
COMMUNITY
BLOG
CONTACT US
SUBSCRIBE
Back to Blog
Matt Butcher
9 Nov 2015
Introducing Helm, The Kubernetes Package Manager
Tags: Helm, Kubernetes
We at Deis are really excited about Kubernetes. In fact, we're hard at work building Deis v2 on top of a Kubernetes base. During this integration, we developed a tool that we think seasoned Kubernetes users will enjoy, and newcomers can use as an onramp for running containerized applications. And today, we're thrilled to announce this new tool.
We call it Helm, and it's the first package manager for Kubernetes.
Inspired by Homebrew, apt, and npm, Helm is a tool for working with Kubernetes-powered applications. It works like this:
A Helm package is bundled up as a chart.
The charts are collected together in a repository that you can search. Helm uses git under the hood for storing and organizing chart data.
Using the helm tool, you can find, customize, manage and install these charts.
Helm makes it easy run apps and services inside Kubernetes.
HELM'S GOALS

Why Helm 4?

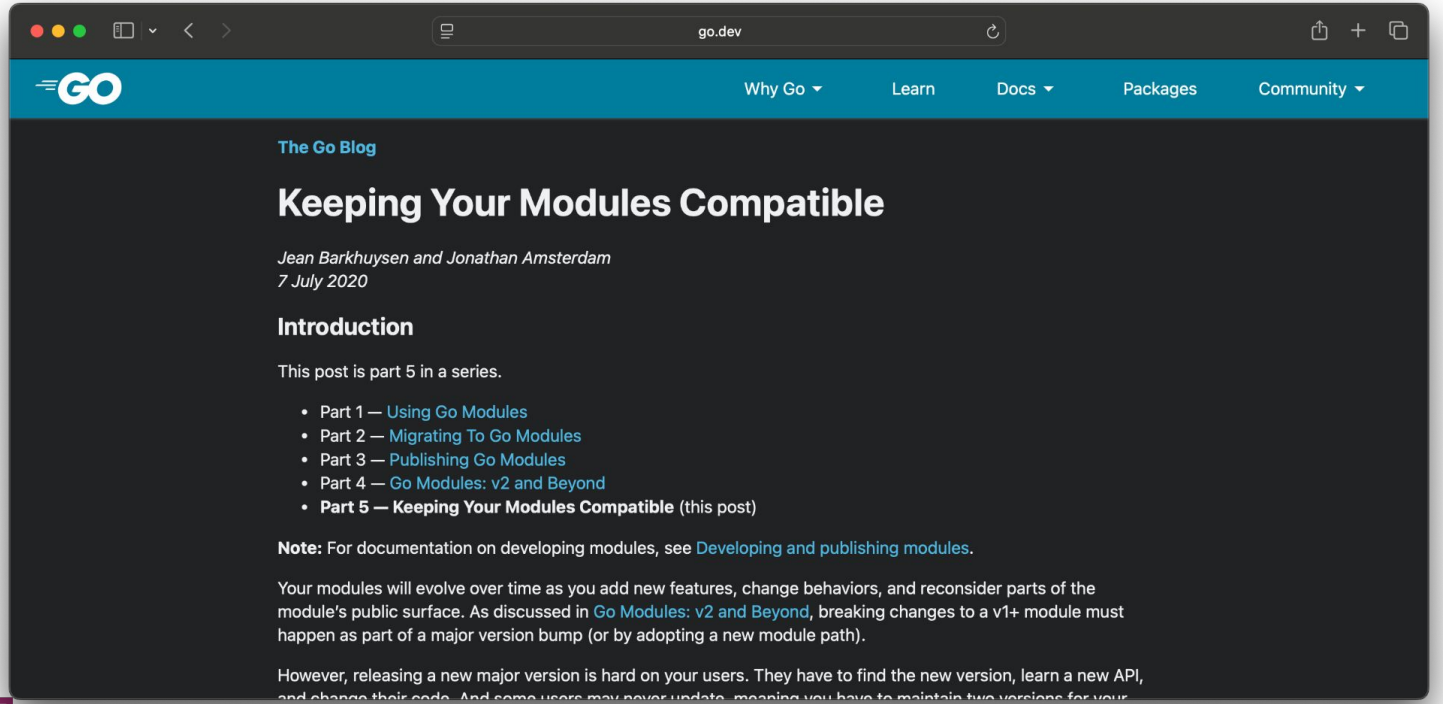
Helm At 10 Years Old



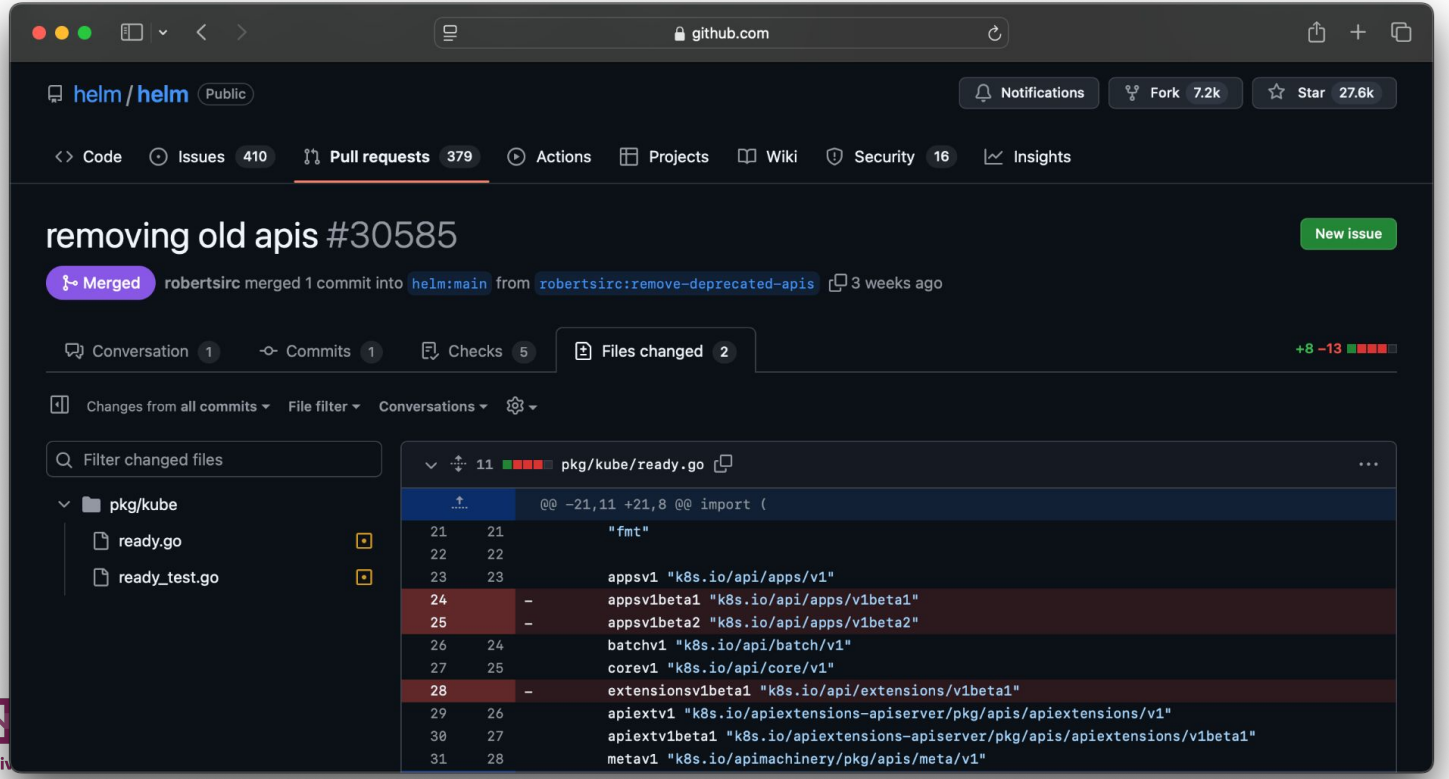
Ugly & Separated Interfaces



Following Go Language Guidance



Helm Support ***OLD*** Kubernetes



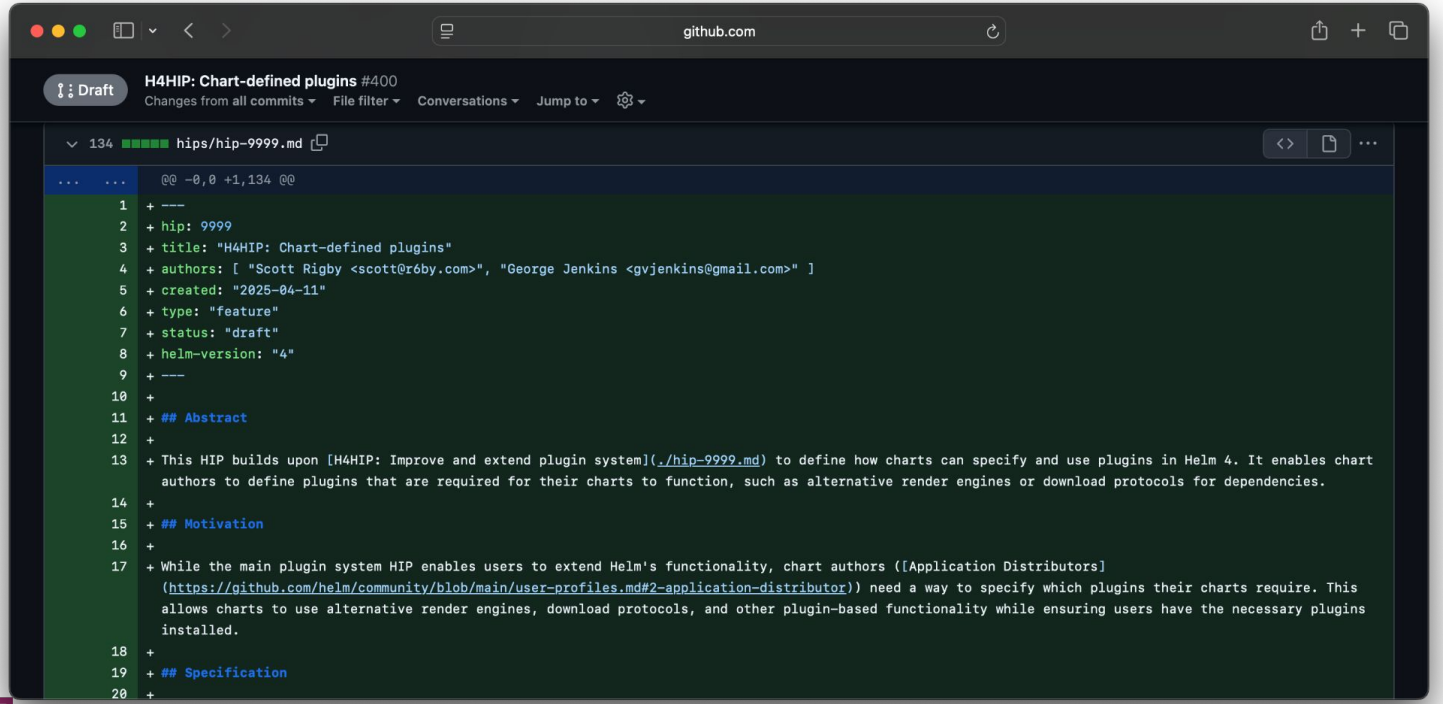
Logging

Helm v3 has simple logging that predated the popularity of logrus and other loggers.

```go
func Debug(format string, v ...interface{}) {
    if settings.Debug {
        log.Output(2, fmt.Sprintf("[debug] "+format+"\n", v...))
    }
}

func Warning(format string, v ...interface{}) {
    fmt.Fprintf(os.Stderr, "WARNING: "+format+"\n", v...)
}
```

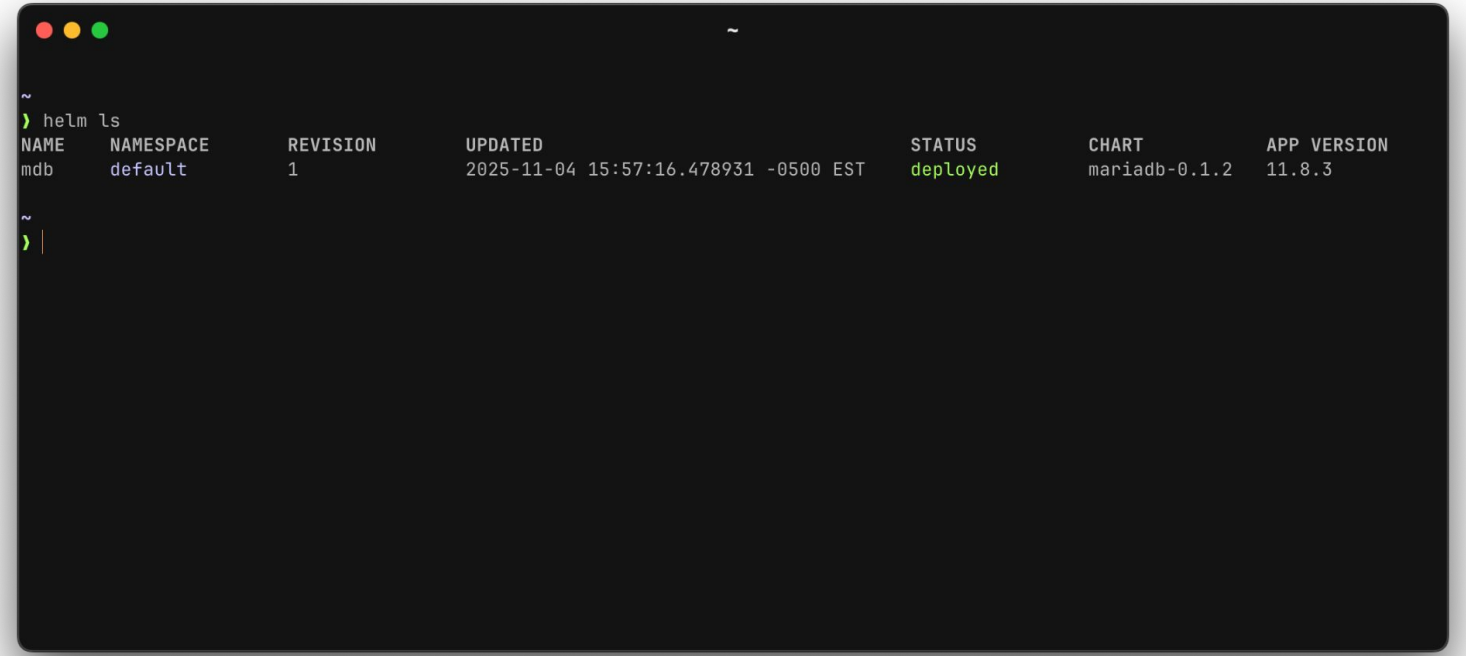
```go
            if errOutputting := cfg.outputLogsByPolicy(h, rl.Namespace, release.HookOutputOnFailed); errOutputting != nil {
                // We log the error here as we want to propagate the hook failure upwards to the release object.
                log.Printf("error outputting logs for hook failure: %v", errOutputting)
            }
            // If a hook is failed, check the annotation of the hook to determine whether the hook should be deleted
            // under failed condition. If so, then clear the corresponding resource object in the hook
            if errDeleting := cfg.deleteHookByPolicy(h, release.HookFailed, timeout); errDeleting != nil {
                // We log the error here as we want to propagate the hook failure upwards to the release object.
                log.Printf("error deleting the hook resource on hook failure: %v", errDeleting)
            }
```

Hard To Implement Chart Features

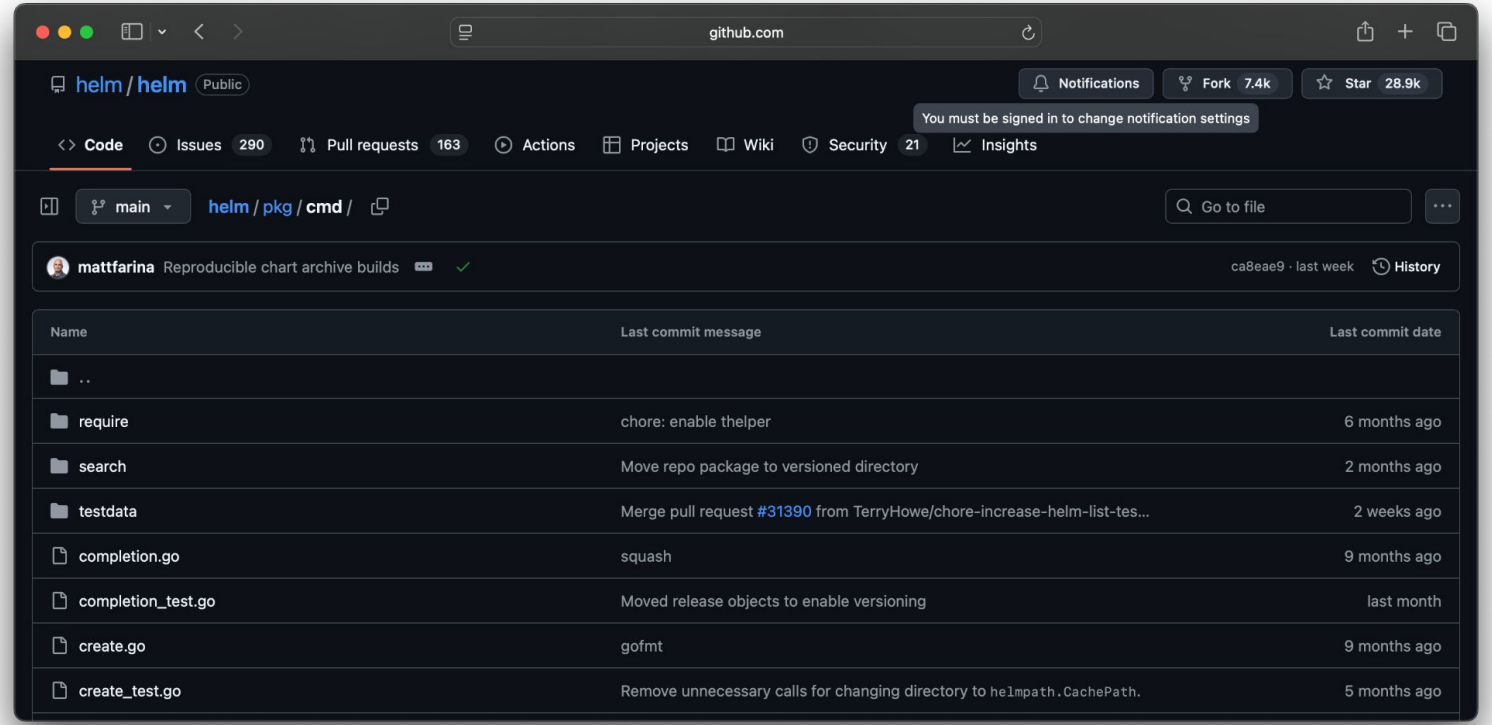


What Changed In Helm v4

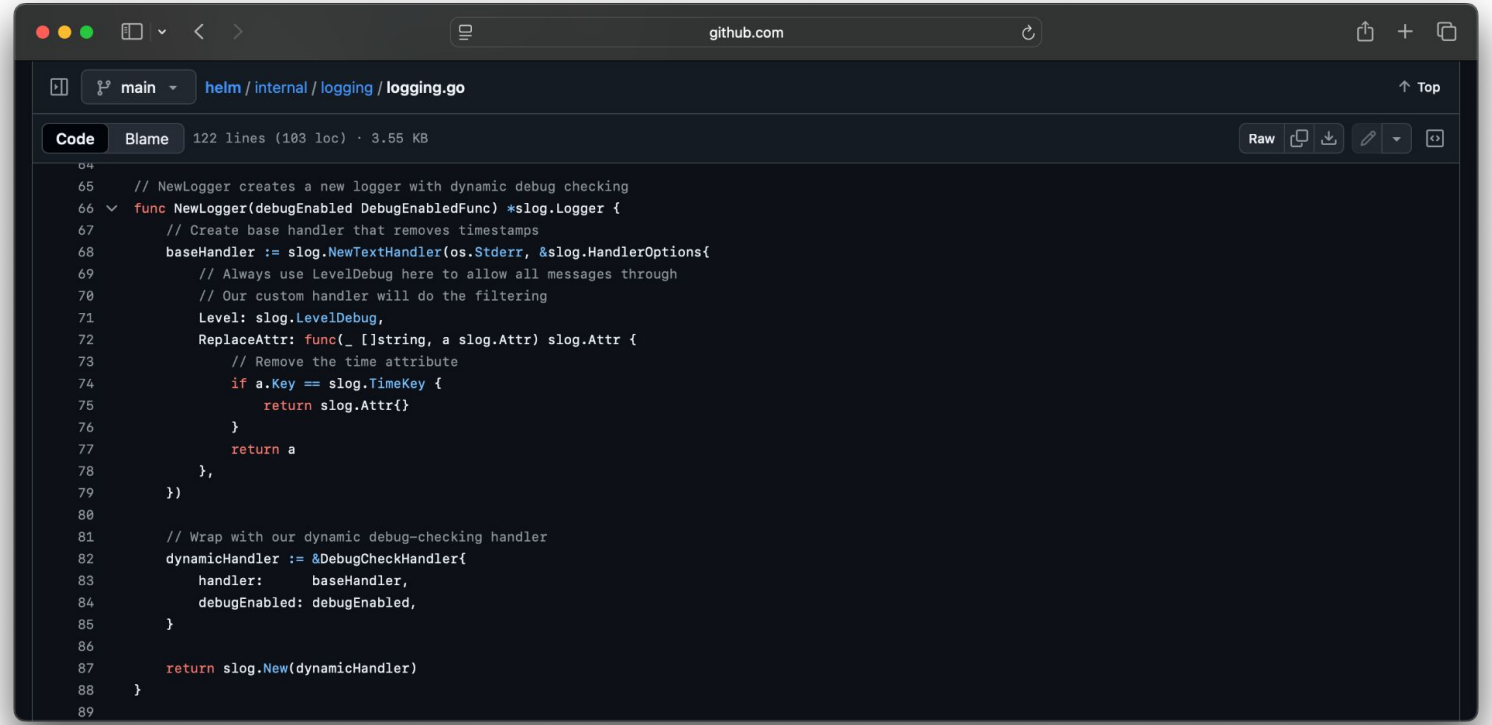
We Have Color



Commands Can Be Embedded In Other Apps



slog Based Logging




```go
// NewLogger creates a new logger with dynamic debug checking
func NewLogger(debugEnabled DebugEnabledFunc) *slog.Logger {
    // Create base handler that removes timestamps
    baseHandler := slog.NewTextHandler(os.Stderr, &slog.HandlerOptions{
        // Always use LevelDebug here to allow all messages through
        // Our custom handler will do the filtering
        Level: slog.LevelDebug,
        ReplaceAttr: func(_ []string, a slog.Attr) slog.Attr {
            // Remove the time attribute
            if a.Key == slog.TimeKey {
                return slog.Attr{}
            }
            return a
        },
    })

    // Wrap with our dynamic debug-checking handler
    dynamicHandler := &DebugCheckHandler{
        handler:      baseHandler,
        debugEnabled: debugEnabled,
    }

    return slog.New(dynamicHandler)
}
```

Reproducible Chart Builds

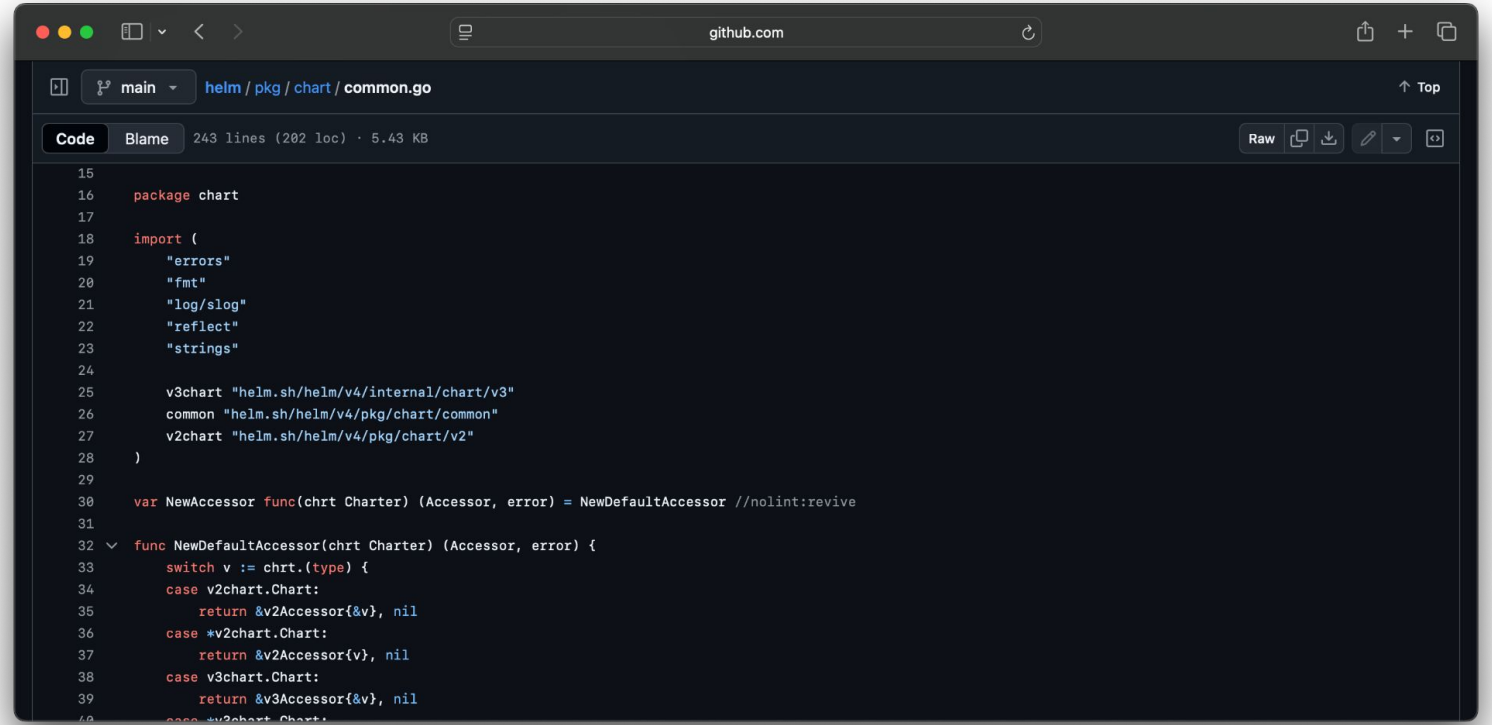
+

Reproducible Builds

Helm Plugins Built With Web Assembly

+

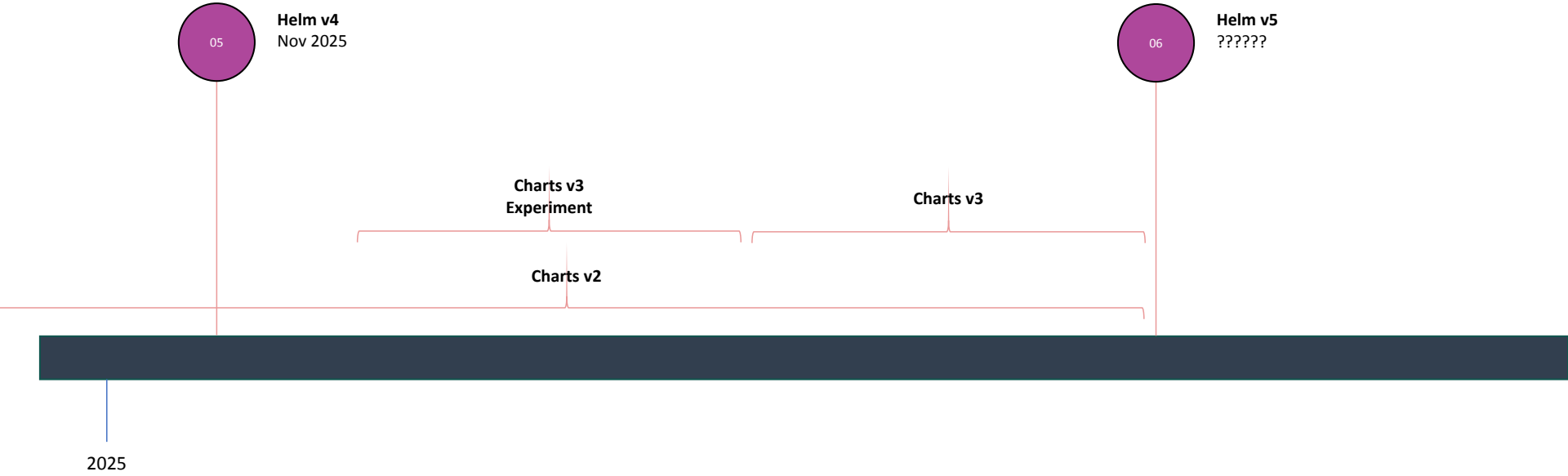
Groundwork For New Chart apiVersions




```go
package chart

import (
    "errors"
    "fmt"
    "log/slog"
    "reflect"
    "strings"

    v3chart "helm.sh/helm/v4/internal/chart/v3"
    common "helm.sh/helm/v4/pkg/chart/common"
    v2chart "helm.sh/helm/v4/pkg/chart/v2"
)

var NewAccessor func(chrt Charter) (Accessor, error) = NewDefaultAccessor //nolint:revive

func NewDefaultAccessor(chrt Charter) (Accessor, error) {
    switch v := chrt.(type) {
    case v2chart.Chart:
        return &v2Accessor{&v}, nil
    case *v2chart.Chart:
        return &v2Accessor{v}, nil
    case v3chart.Chart:
        return &v3Accessor{&v}, nil
```

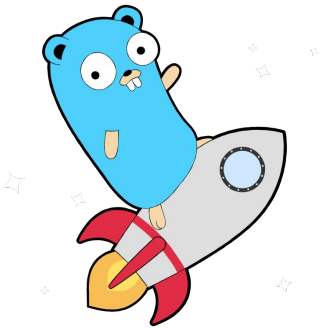
What Is Still To Come In Helm v4?

Charts v3 Experiment Timeline



SDK Improvements

goreleaser + cosign



+

sigstore

cosign

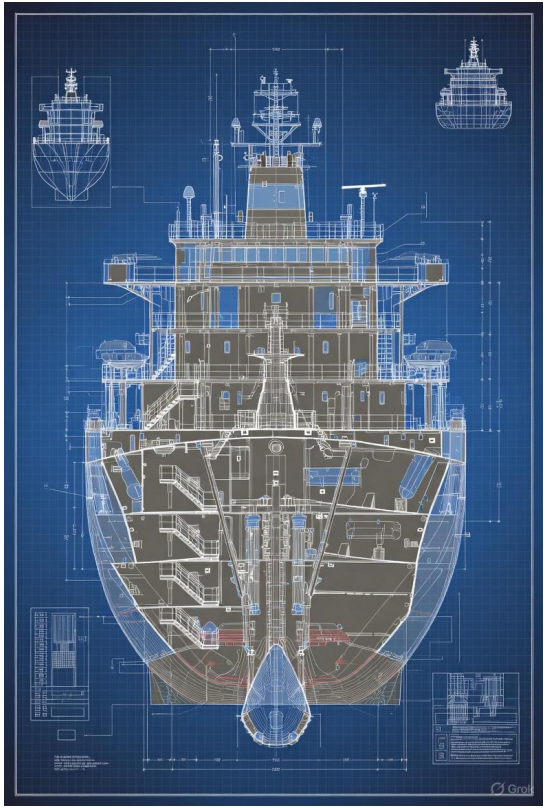
How Long Will v3 Be Supported?

Helm v3 Support Cycle

- Bug fixes until July 8th 2026.
- Security fixes until November 11th 2026.
- No feature backports

Helm v5

- When: Sometime in the future (years away)
- Why: Breaking changes are needed to enable the future of Helm
- Ideas: ???



Getting Involved

Collaborate on Slack

Engage with other members of the Helm community

Code Contribution

Development of core capabilities, documentation and other assets

Join the Helm Community Meetings

Participate with other members of the Helm community to discuss efforts related to Helm

Getting Involved

Charts Tooling

Testing, CI actions, and more

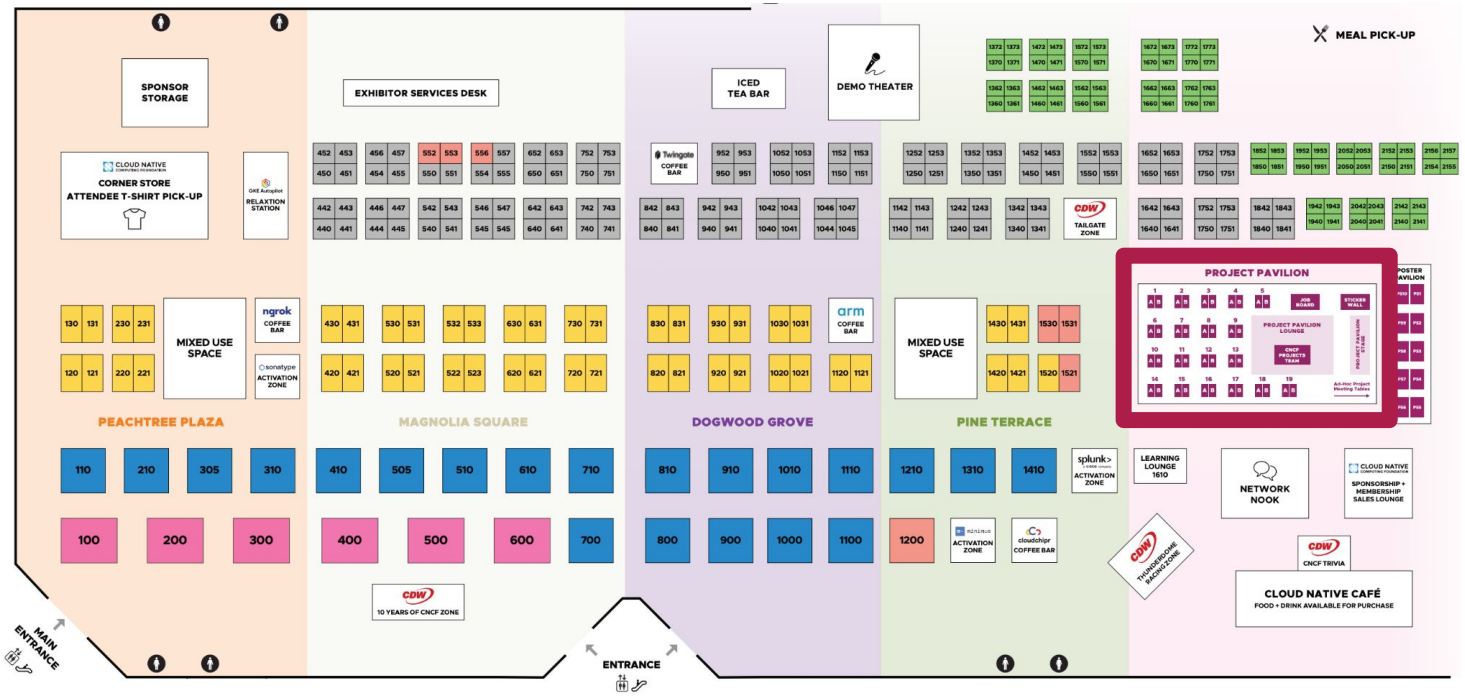
Documentation

The website, docs, and translations

Helm CLI & SDK

Work on Helm itself. The Client and the SDK used by other applications such as Flux

Find Us In The Project Pavilion (afternoons)



Q&A


KubeCon
CloudNativeCon
North America 2025