Practical Go

Shubham Sharma

Software Engineer, Microsoft

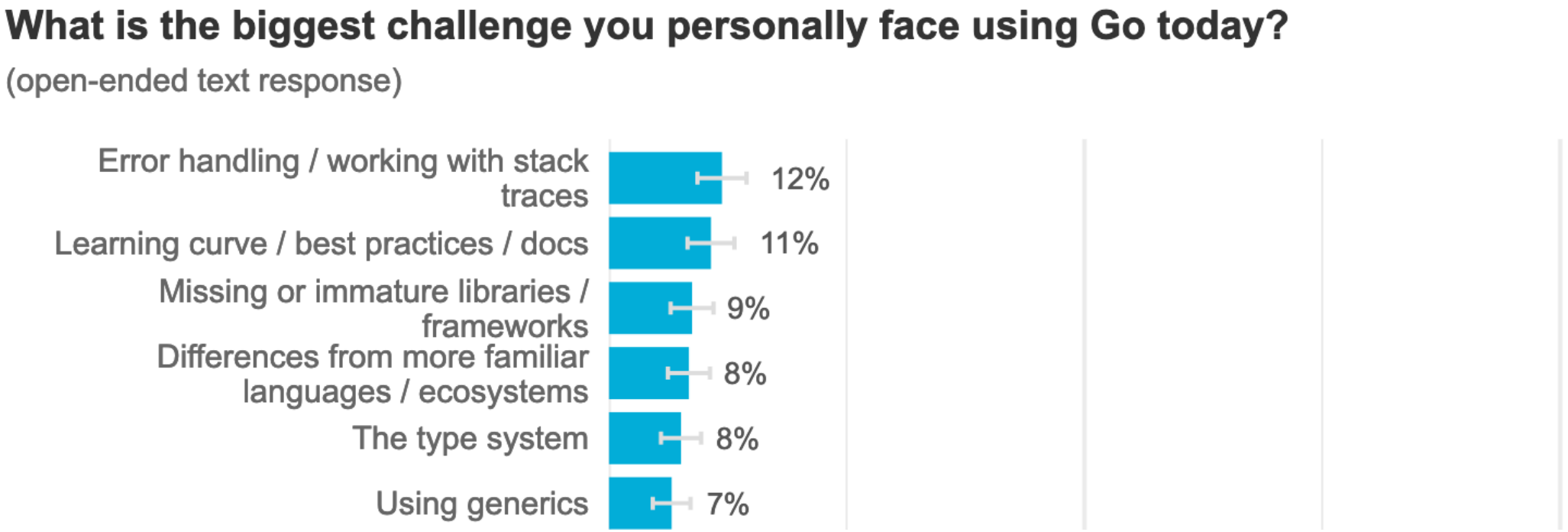
Agenda

Practical examples of using Go in the wild.

- Errors in Go
- Context, Timeouts, and Cancellations
- Concurrency patterns
- Structuring a Go project

Errors in Go

to, err := human()



https://go.dev/blog/survey2023-q1-results

Error handling philosophies

Single return languages in C

Throwing exceptions in C++

Checked exceptions in Java

Returning exceptions in Go

Error handling philosophies

Single return languages in C

Throwing exceptions in C++

Checked exceptions in Java

Returning exceptions in Go

Error handling philosophies

Single return languages in C

Throwing exceptions in C++

Checked exceptions in Java

Returning exceptions in Go

Error handling philosophies

Single return languages in C

Throwing exceptions in C++

Checked exceptions in Java

Returning exceptions in Go

Errors in Go

- Return error as value
- Always inspect error before using other return values

```
type error interface {
    Error() string
}
```

```
// src: strconv stdlib
// Converting a string to integer
func Atoi(s string) (int, error)

n, err := strconv.Atoi("100")
if err != nil {
    log.Fatal(err)
}
// Do something with n
```

Peeking into the standard library

```
57 package errors
58
59 // New returns an error that formats as the given text.
60 // Each call to New returns a distinct error value even if the text is identical.
61 func New(text string) error {
62   return &errorString{text}
63 }
64
65 // errorString is a trivial implementation of error.
66 type errorString struct {
67   s string
68 }
69
70 func (e *errorString) Error() string {
71   return e.s
72 }
```

https://cs.opensource.google/go/go/+/refs/tags/go1.21.5:src/errors/errors.go

Returning Errors

```
func Sqrt(f float64) (float64, error) {
    if f < 0 {
        return 0, errors.New("math: square root of negative number")
    }
    // implementation
}
```

```
func Sqrt(f float64) (float64, error) {
    if f < 0 {
        return 0, fmt.Errorf("math: square root of negative number %g", f)
    }
    // implementation
}
```

Sentinel Errors

```
var ErrSqrtNegativeNumber = errors.New("math: square root of negative number")

func Sqrt(f float64) (float64, error) {
  if f < 0 {
    return 0, ErrSqrtNegativeNumber
  }
  // implementation
}
```

```
  x := 0
  y, err := Sqrt(f)
  if err != nil {
      switch {
      case errors.Is(err, ErrSqrtNegativeNumber):
          fmt.Println("square root of negative number error")
      default:
          fmt.Printf("unexpected sqrt error: %s\n", err)
      }
      return
    }
```

In the wild…

```
69    // IsErrorPermanent returns true if `err` should be treated as a
70    // permanent error that cannot be retried.
71    func IsErrorPermanent(err error) bool {
72            return errors.Is(err, ErrOpenState) || errors.Is(err, ErrTooManyRequests)
73    }
```

https://github.com/dapr/dapr/blob/master/pkg/resiliency/breaker/circuitbreaker.go#L71

In the wild…

```
27    // Validate validates the common rules for all requests.
28    func Validate(_ context.Context, req *sentryv1pb.SignCertificateRequest) (spiffeid.TrustDomain, bool, error) {
29            err := errors.Join(
30                    validation.ValidateSelfHostedAppID(req.GetId()),
31                    appIDLessOrEqualTo64Characters(req.GetId()),
32                    csrIsRequired(req.GetCertificateSigningRequest()),
33                    namespaceIsRequired(req.GetNamespace()),
34            )
35            if err != nil {
36                    return spiffeid.TrustDomain{}, false, fmt.Errorf("invalid request: %w", err)
37            }
```

https://github.com/dapr/dapr/blob/master/pkg/sentry/server/validator/internal/common.go#L28

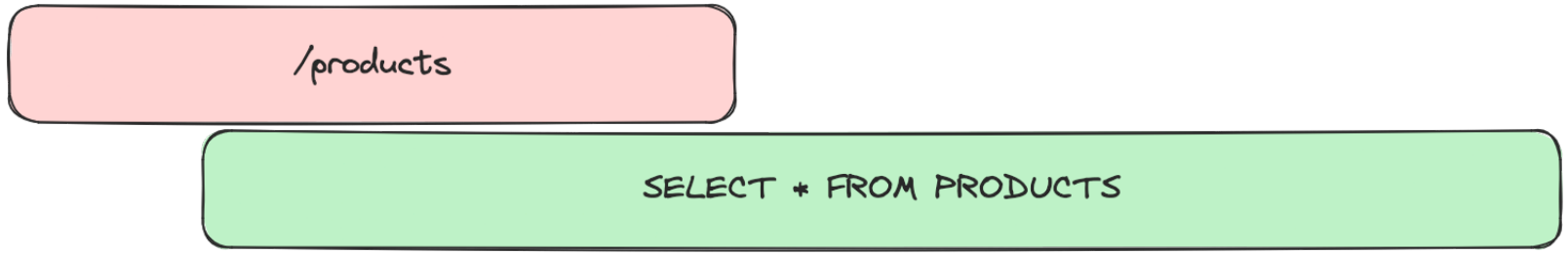
Context, Timeouts, and Cancellations

Scenario 1

Fetching all products
using an API call

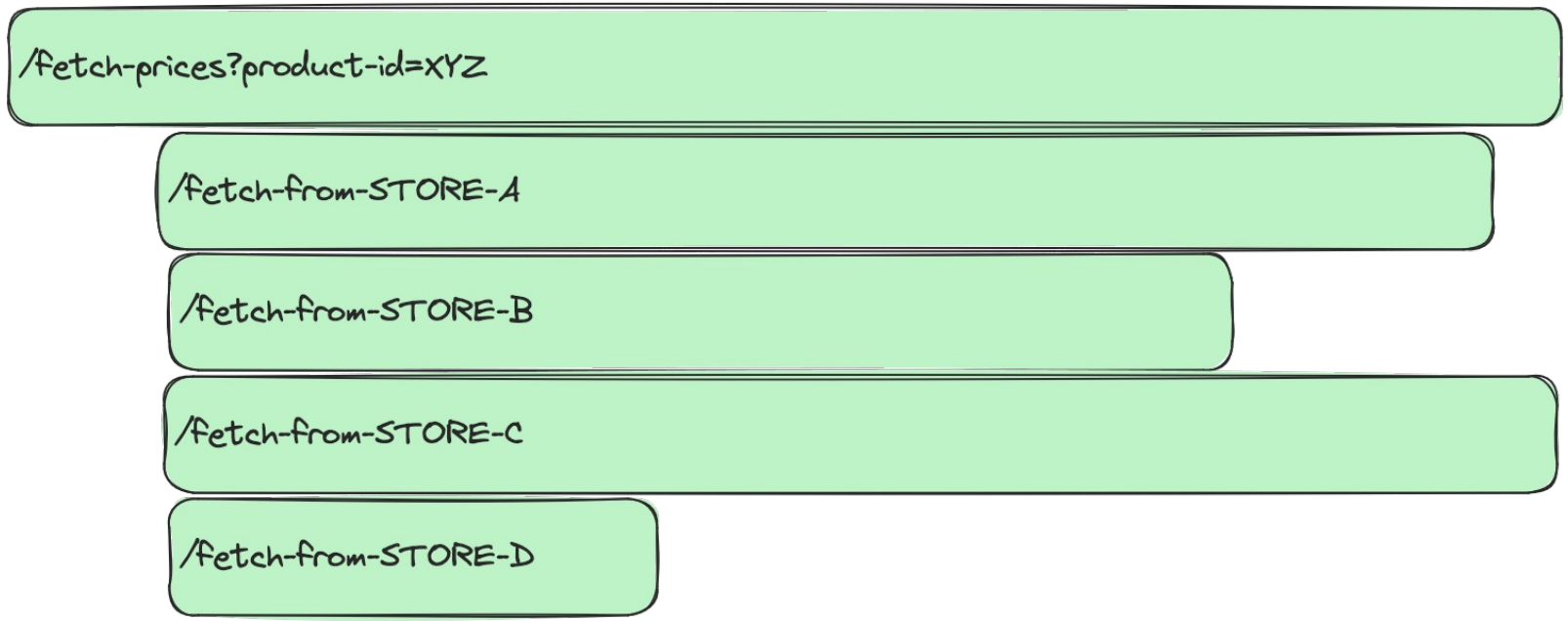
Scenario 1

User cancels the request midway



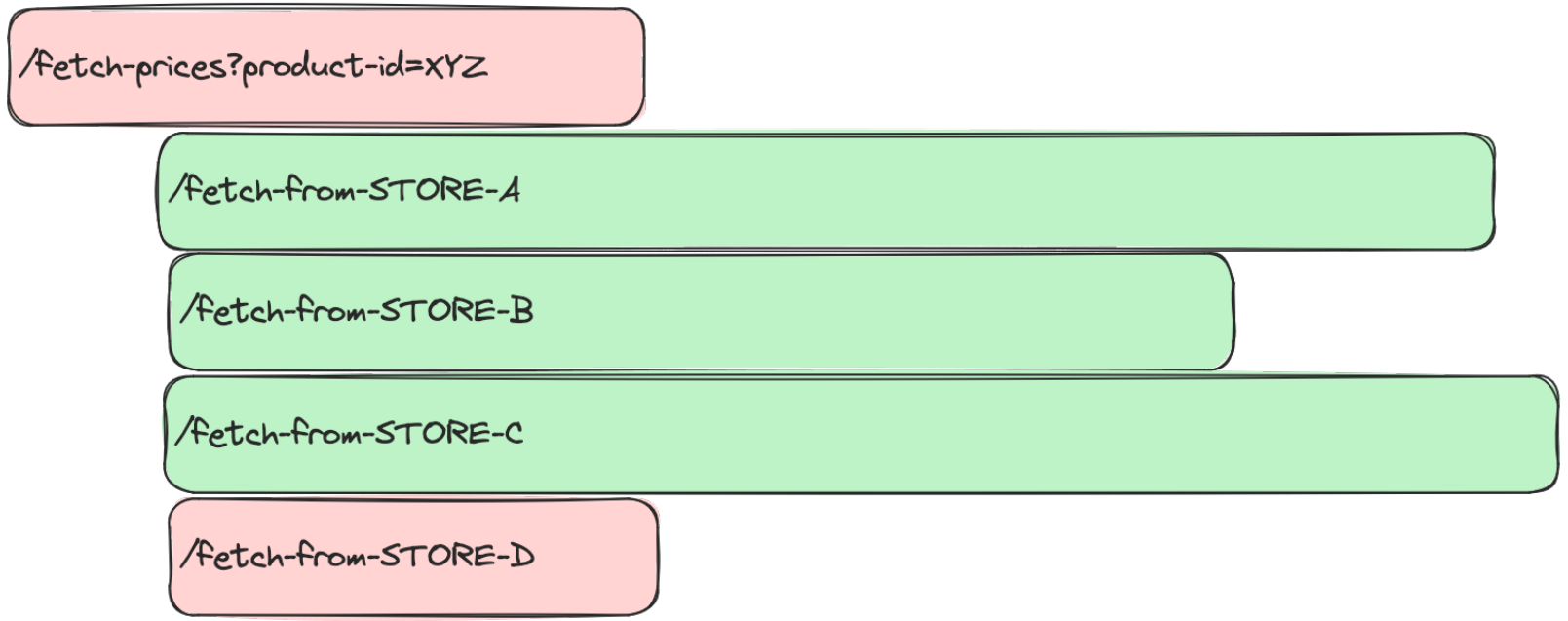
Scenario 2

Concurrent API calls



Scenario 2

One API call fails



The context package

Overview

Package context defines the Context type, which carries deadlines, cancellation signals, and other request-scoped values across API boundaries and between processes.

Incoming requests to a server should create a Context, and outgoing calls to servers should accept a Context. The chain of function calls between them must propagate the Context, optionally replacing it with a derived Context created using WithCancel, WithDeadline, WithTimeout, or WithValue. When a Context is canceled, all Contexts derived from it are also canceled.

https://pkg.go.dev/context#pkg-overview

Context

```
func main() {
    http.HandleFunc("/products", handleProducts)
    http.ListenAndServe(":8080", nil)
}

func handleProducts(w http.ResponseWriter, r *http.Request) {
    ps, err := getProducts()
    // ...
}

func getProducts() ([]Product, error) {
    db := getDB()

    var ps []Product
    if err := db.Find(&ps).Error; err != nil {
        return nil, err
    }

    return ps, nil
}
```

Context

```
func (*http.Request).Context() context.Context
```

Context returns the request's context. To change the context, use Clone or WithContext.

The returned context is always non-nil; it defaults to the background context.

For outgoing client requests, the context controls cancellation.

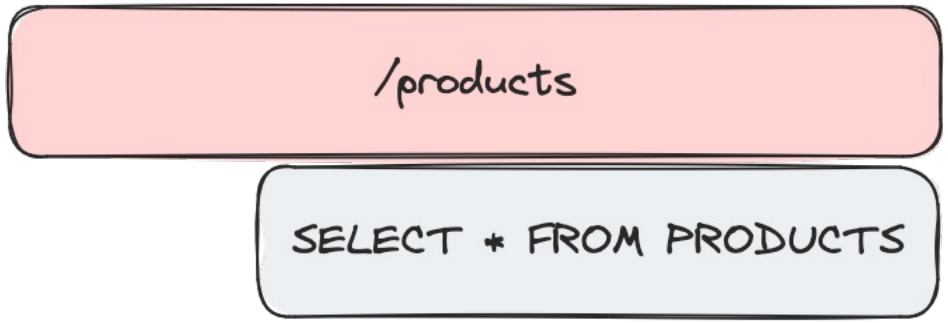
For incoming server requests, the context is canceled when the client's connection closes, the request is canceled (with HTTP/2), or when the ServeHTTP method returns.

(http.Request).Context on pkg.go.dev

```go
func main() {
    http.HandleFunc("/products", handleProducts)
    http.ListenAndServe(":8080", nil)
}

func handleProducts(w http.ResponseWriter, r *http.Request) {
    ps, err := getProducts()
    // ...
}

func getProducts() ([]Product, error) {
    db := getDB()

    var ps []Product
    if err := db.Find(&ps).Error; err != nil {
        return nil, err
    }

    return ps, nil
}
```

```go
func handleProducts(w http.ResponseWriter, r *http.Request) {
    // Pass the request context to the database layer
    ps, err := getProducts(r.Context())
    // ...
}

func getProducts(ctx context.Context) ([]Product, error) {
    db := getDB()

    var ps []Product
    // Bind the database operation to a context.
    // Returns an error if the context is cancelled midway.
    if err := db.WithContext(ctx).Find(&ps).Error; err != nil {
        return nil, err
    }

    return ps, nil
}
```

Context


/products
SELECT * FROM PRODUCTS


/fetch-prices?product-id=XYZ
/fetch-from-STORE-A
/fetch-from-STORE-B
/fetch-from-STORE-C
/fetch-from-STORE-D

Timeouts

```
var context.DeadlineExceeded error
```

DeadlineExceeded is the error returned by [Context.Err] when the context's deadline passes.

context.DeadlineExceeded on pkg.go.dev

```go
func longRunningTask(ctx context.Context) (err error) {
    select {
    case <-time.After(5 * time.Second):
        fmt.Println("Long running task finished")
        return nil
    case <-ctx.Done():
        fmt.Println("Context is done")
        return ctx.Err()
    }
}
```

```go
// Create a context with a three seconds timeout.
ctx, cancel := context.WithTimeout(context.Background(), 3*time.Second)
defer cancel()

err := longRunningTask(ctx)
switch err {
case nil:
    fmt.Println("No error")
case context.DeadlineExceeded:
    fmt.Println("Deadline exceeded")
default:
    fmt.Println("Other error")
}
```

In the wild…

```
242             select {
243             case <-ctx.Done():
244                     log.Info("Sidecar injector is shutting down")
245                     shutdownCtx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
246                     defer cancel()
247                     if err := i.server.Shutdown(shutdownCtx); err != nil {
248                             return fmt.Errorf("error while shutting down injector: %v; %v", err, <-errCh)
249                     }
250                     return <-errCh
251             case err := <-errCh:
252                     return err
253             }
```

https://github.com/dapr/dapr/blob/master/pkg/injector/service/injector.go#L216

Cancellations

```go
func worker(ctx context.Context) {
    for {
        select {
        case <-ctx.Done():
            fmt.Println("Worker: Received cancellation signal. Exiting...")
            return
        default:
            // Simulate some work
            fmt.Println("Worker: Performing some work...")
            time.Sleep(2 * time.Second)
        }
    }
}
```

```go
// Create a context with cancellation function
ctx, cancel := context.WithCancel(context.Background())

// Start a worker goroutine with the created context
go worker(ctx)

// Simulate main program execution for a while
time.Sleep(5 * time.Second)

// Cancel the context to signal the worker to stop
fmt.Println("Main: Cancelling the context.")
cancel()
```

In the wild…

```
350     // checkShouldLead returns true if the caller should attempt leader election for a remote cluster.
351     func (m *Multicluster) checkShouldLead(client kubelib.Client, systemNamespace string, stop <-chan struct{}) bool {
352             var res bool
353             if features.ExternalIstiod {
354                     b := backoff.NewExponentialBackOff(backoff.DefaultOption())
355                     ctx, cancel := context.WithCancel(context.Background())
356                     go func() {
357                             select {
358                             case <-stop:
359                                     cancel()
360                             case <-ctx.Done():
361                             }
362                     }()
363                     defer cancel()
```

https://github.com/istio/istio/blob/master/pilot/pkg/serviceregistry/kube/controller/multicluster.go#L351

Values inside context

Useful when values are scoped to an operation.

```
type ctxKey string

var CtxKeyEvent = ctxKey("Event")

func main() {
    ctx := context.WithValue(context.Background(), CtxKeyEvent, "Go Bootcamp")
    printEvent(ctx)
}

func printEvent(ctx context.Context) {
    value := ctx.Value(CtxKeyEvent)
    if value != nil {
        fmt.Printf("Event: %s\n", value)
        return
    }
    fmt.Print("Unable to find the context value")
}
```

In the wild…

```
62    // RequestID is a middleware that injects a request ID into the context of each
63    // request. A request ID is a string of the form "host.example.com/random-0001",
64    // where "random" is a base62 random string that uniquely identifies this go
65    // process, and where the last number is an atomically incremented request
66    // counter.
67    func RequestID(next http.Handler) http.Handler {
68            fn := func(w http.ResponseWriter, r *http.Request) {
69                    ctx := r.Context()
70                    requestID := r.Header.Get(RequestIDHeader)
71                    if requestID == "" {
72                            myid := atomic.AddUint64(&reqid, 1)
73                            requestID = fmt.Sprintf("%s-%06d", prefix, myid)
74                    }
75                    ctx = context.WithValue(ctx, RequestIDKey, requestID)
76                    next.ServeHTTP(w, r.WithContext(ctx))
77            }
78            return http.HandlerFunc(fn)
79    }
```

https://github.com/go-chi/chi/blob/master/middleware/request_id.go#L67

Concurrency patterns

Concurrency is not parallelism

https://go.dev/blog/waza-talk

- Concurrency is not parallelism, although it enables parallelism.
- If you have only one processor, your program can still be concurrent, but it cannot be parallel.
- On the other hand, a well-written concurrent program might run efficiently in parallel on a multiprocessor.

Concurrency is not parallelism

runtime.GOMAXPROCS

- Set to number of cores on the machine by default

Common concurrency patterns

- https://go.dev/talks/2012/concurrency.slide
- https://github.com/lotusirous/go-concurrency-patterns

Common concurrency patterns

- A boring function - https://go.dev/play/p/8RQH-hbjZ7W
- Hello channels - https://go.dev/play/p/amazakVmwFy
- Generators - https://go.dev/play/p/9ykTDe7qLSw
- Timeouts with channels - https://play.golang.org/p/WIqNvmxiYvn
- Stop channels - https://play.golang.org/p/rKYKqMeoFDq
- Pipelines - https://go.dev/play/p/H6yn_c3TI0o
- Fan-in - https://go.dev/play/p/zLJvE8a6k0R

Structuring your project

Project structure guidelines

- Go has some special conventions
  - E.g., `main` package, `internal`, `vendor` directories
- Learnings from popular Go projects - https://github.com/golang-standards/project-layout

Packages refresher

- Every Go program is made of packages
- Executable programs start running in the `main` package

```
package main

func main() {
    fmt.Println("Hello, world!")
}
```

Packages refresher

- You can import standard libraries, local packages, and third-party packages

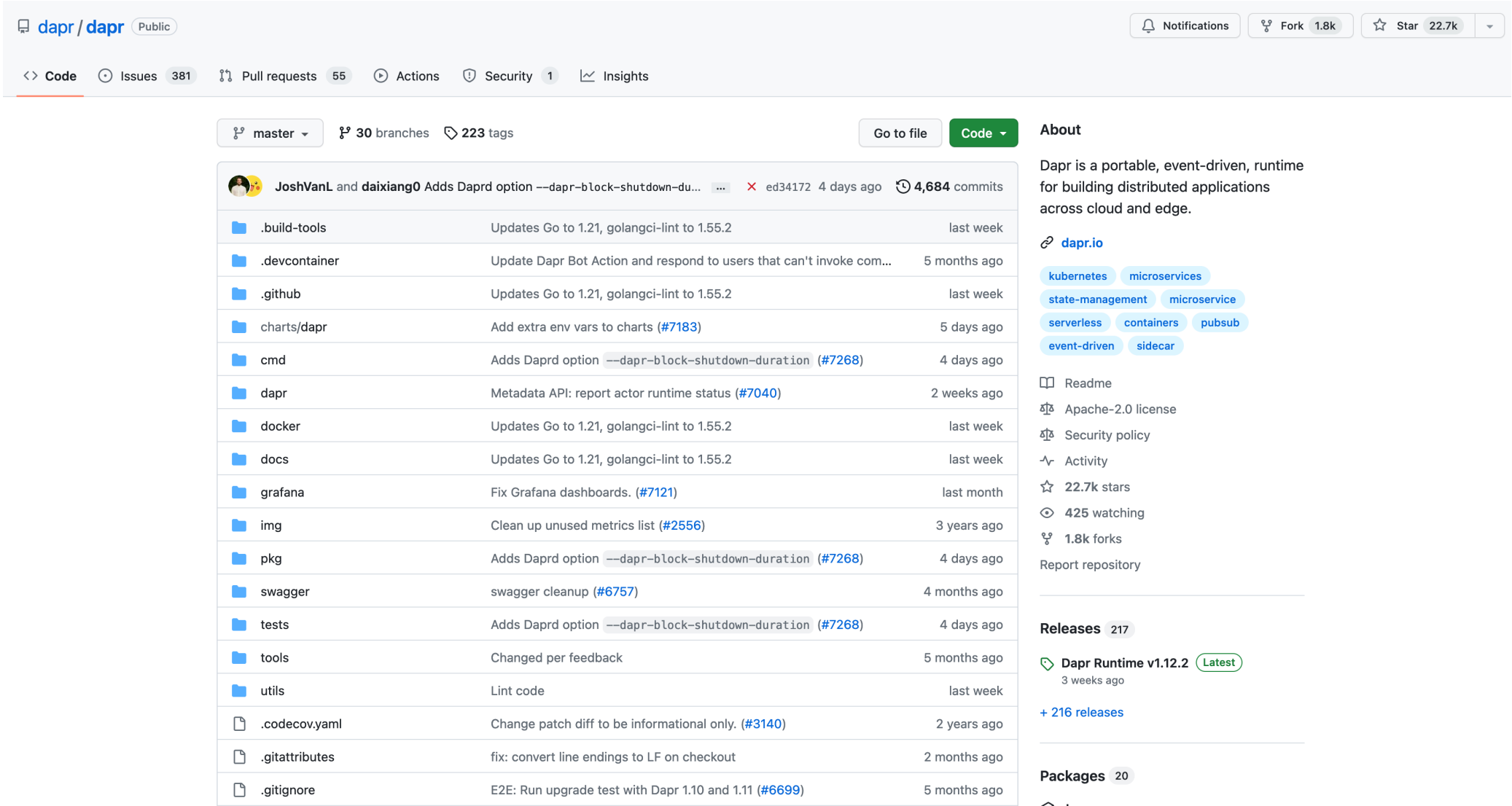
```
package foo

import (
    // standard library packages
    "fmt"
    "math/rand"

    // local packages
    // <module name>/path/to/package
    "mymodule/foo/bar"

    // third party packages
    "github.com/shubham1172/gokv/pkg/store"
)
```

internal

- Special directory name recognized by Go which will prevent one package from being imported by another unless both share a common ancestor
- For example, a package /a/b/c/internal/d/e/f can only be imported by code in the directory tree rooted at /a/b/c. It cannot be imported by code in /a/b/g or in any other repository

https://dave.cheney.net/2019/10/06/use-internal-packages-to-reduce-your-public-api-surface

Structuring a project



dapr / dapr Public

Notifications | Fork 1.8k | Star 22.7k

Code | Issues 381 | Pull requests 55 | Actions | Security 1 | Insights

master | 30 branches | 223 tags | Go to file | Code

JoshVanL and daixiang0 Adds Daprd option --dapr-block-shutdown-du... ed34172 4 days ago 4,684 commits

.build-tools	Updates Go to 1.21, golangci-lint to 1.55.2	last week
.devcontainer	Update Dapr Bot Action and respond to users that can't invoke com...	5 months ago
.github	Updates Go to 1.21, golangci-lint to 1.55.2	last week
charts/dapr	Add extra env vars to charts (#7183)	5 days ago
cmd	Adds Daprd option `--dapr-block-shutdown-duration` (#7268)	4 days ago
dapr	Metadata API: report actor runtime status (#7040)	2 weeks ago
docker	Updates Go to 1.21, golangci-lint to 1.55.2	last week
docs	Updates Go to 1.21, golangci-lint to 1.55.2	last week
grafana	Fix Grafana dashboards. (#7121)	last month
img	Clean up unused metrics list (#2556)	3 years ago
pkg	Adds Daprd option `--dapr-block-shutdown-duration` (#7268)	4 days ago
swagger	swagger cleanup (#6757)	4 months ago
tests	Adds Daprd option `--dapr-block-shutdown-duration` (#7268)	4 days ago
tools	Changed per feedback	5 months ago
utils	Lint code	last week
.codecov.yaml	Change patch diff to be informational only. (#3140)	2 years ago
.gitattributes	fix: convert line endings to LF on checkout	2 months ago
.gitignore	E2E: Run upgrade test with Dapr 1.10 and 1.11 (#6699)	5 months ago

About

Dapr is a portable, event-driven, runtime for building distributed applications across cloud and edge.

dapr.io

kubernetes | microservices | state-management | microservice | serverless | containers | pubsub | event-driven | sidecar

Readme
Apache-2.0 license
Security policy
Activity
22.7k stars
425 watching
1.8k forks
Report repository

Releases 217

Dapr Runtime v1.12.2 Latest
3 weeks ago

+ 216 releases

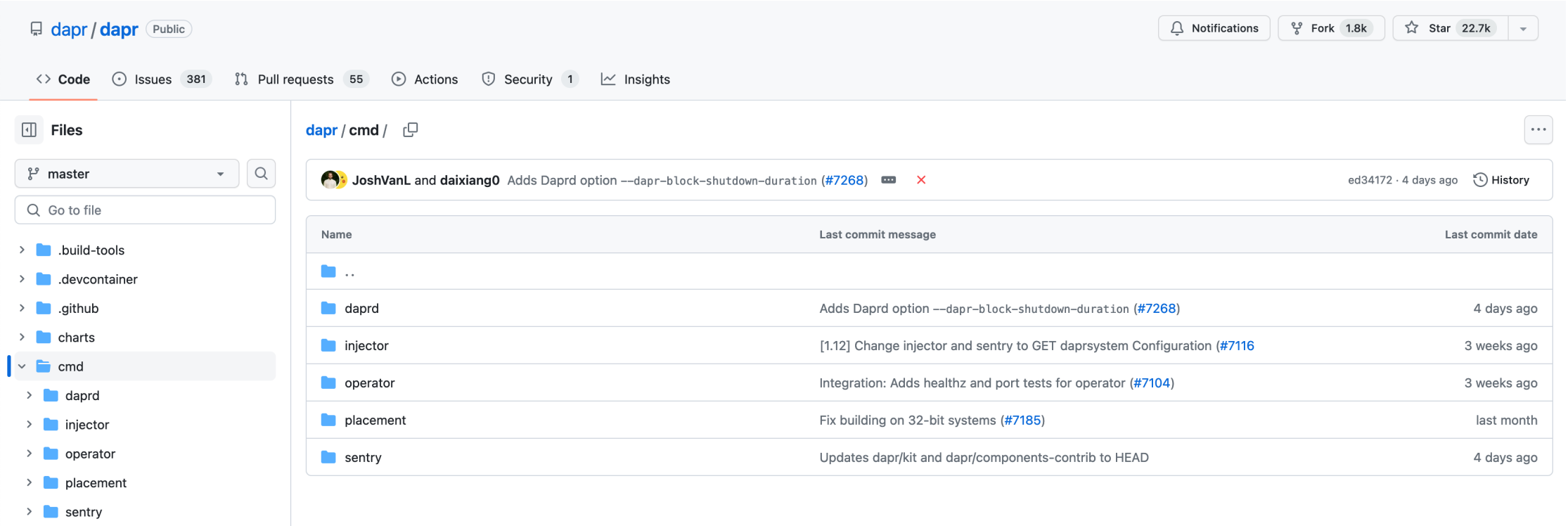
Packages 20

Common directories

- /cmd
  - The directory name for each application should match the name of the executable you want to have.
  - Does not contain a lot of code, mostly flags and arguments for the executable
- /internal
- /pkg
- /vendor

Common directories

/cmd

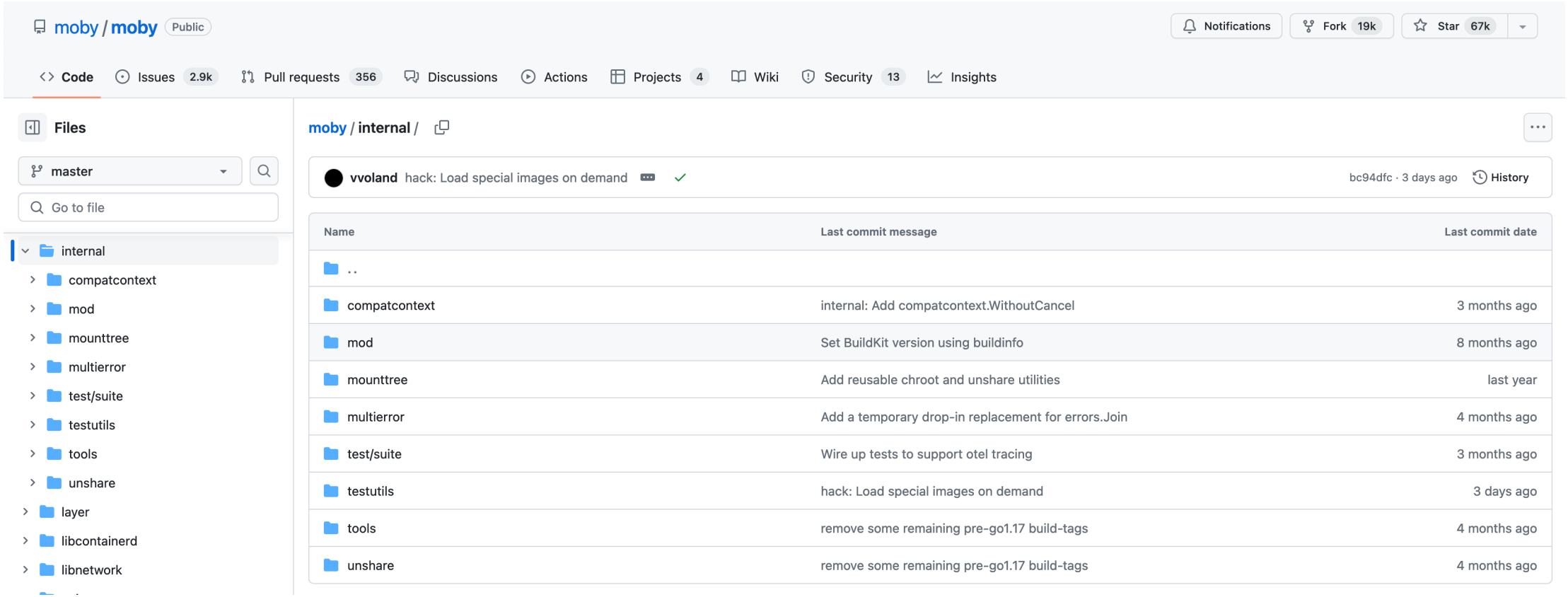


Common directories

- /cmd
- /internal
  - Private packages, cannot be imported by others in any library or application.
  - Enforced by the compiler
- /pkg
- /vendor

Common directories

/internal

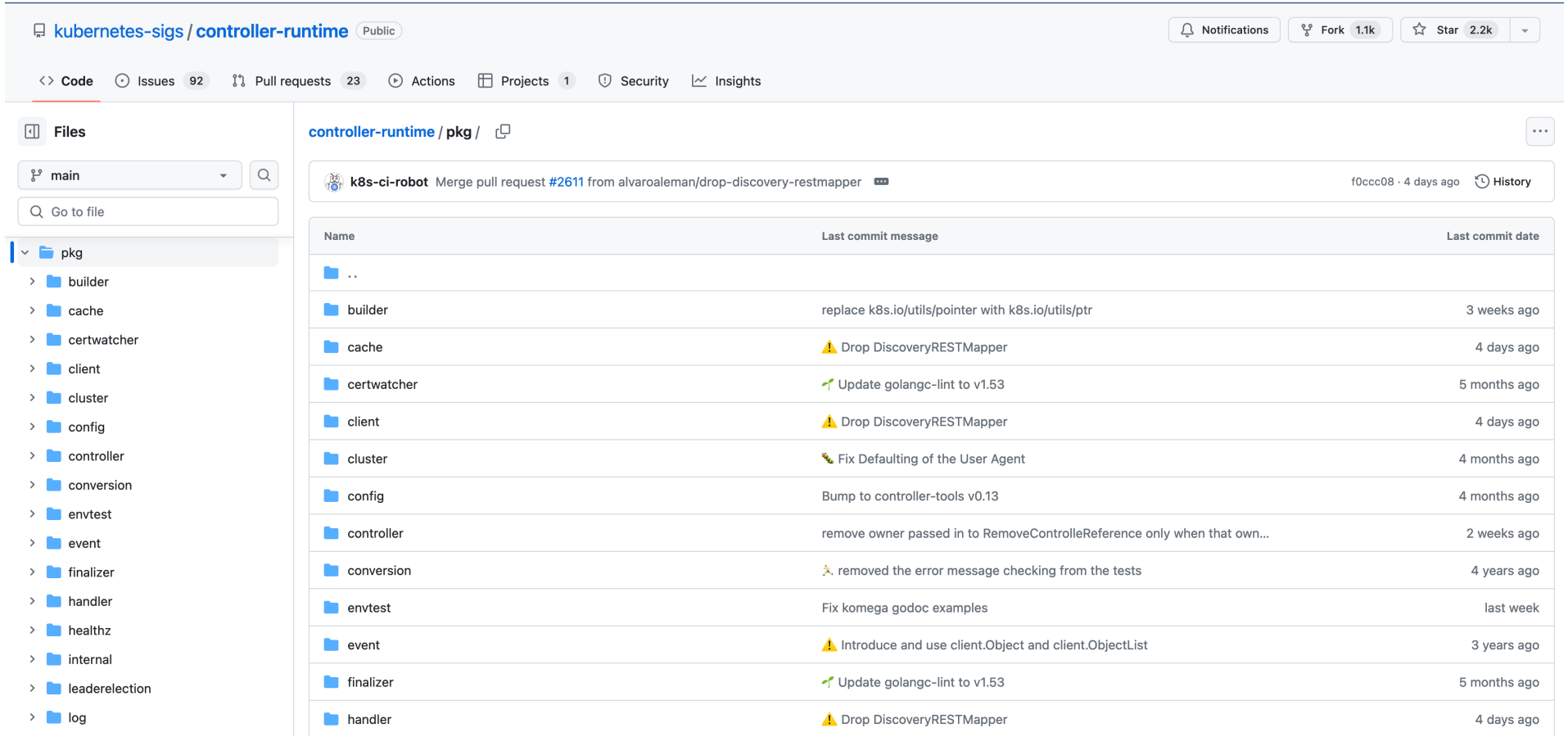


Common directories

- /cmd
- /internal
- /pkg
  - Most other projects will look to import what’s in here
  - Good way to explicitly communicate that the code is safe for use by others
  - Not universal but extremely popular
- /vendor

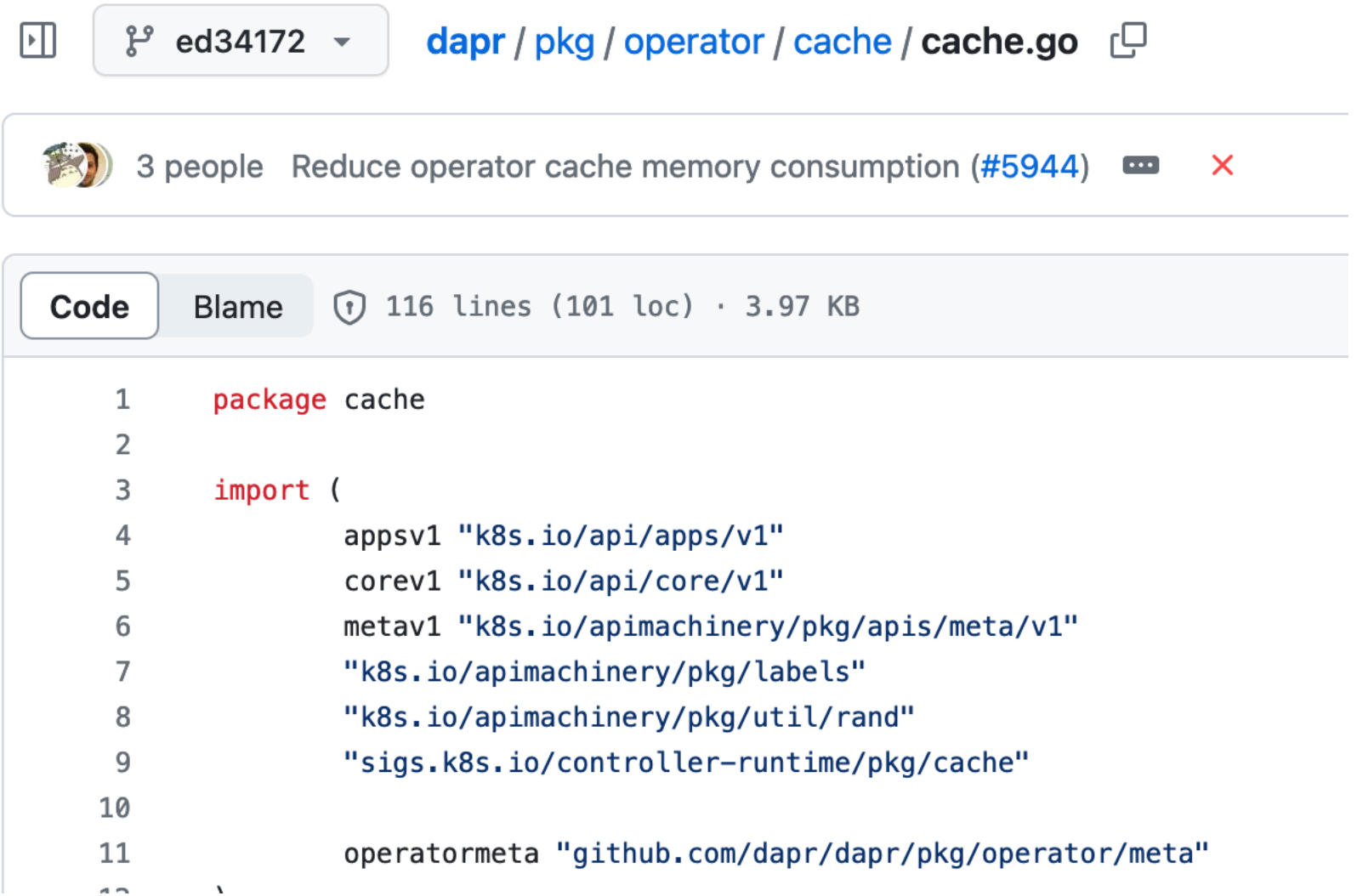
Common directories

/pkg



Common directories

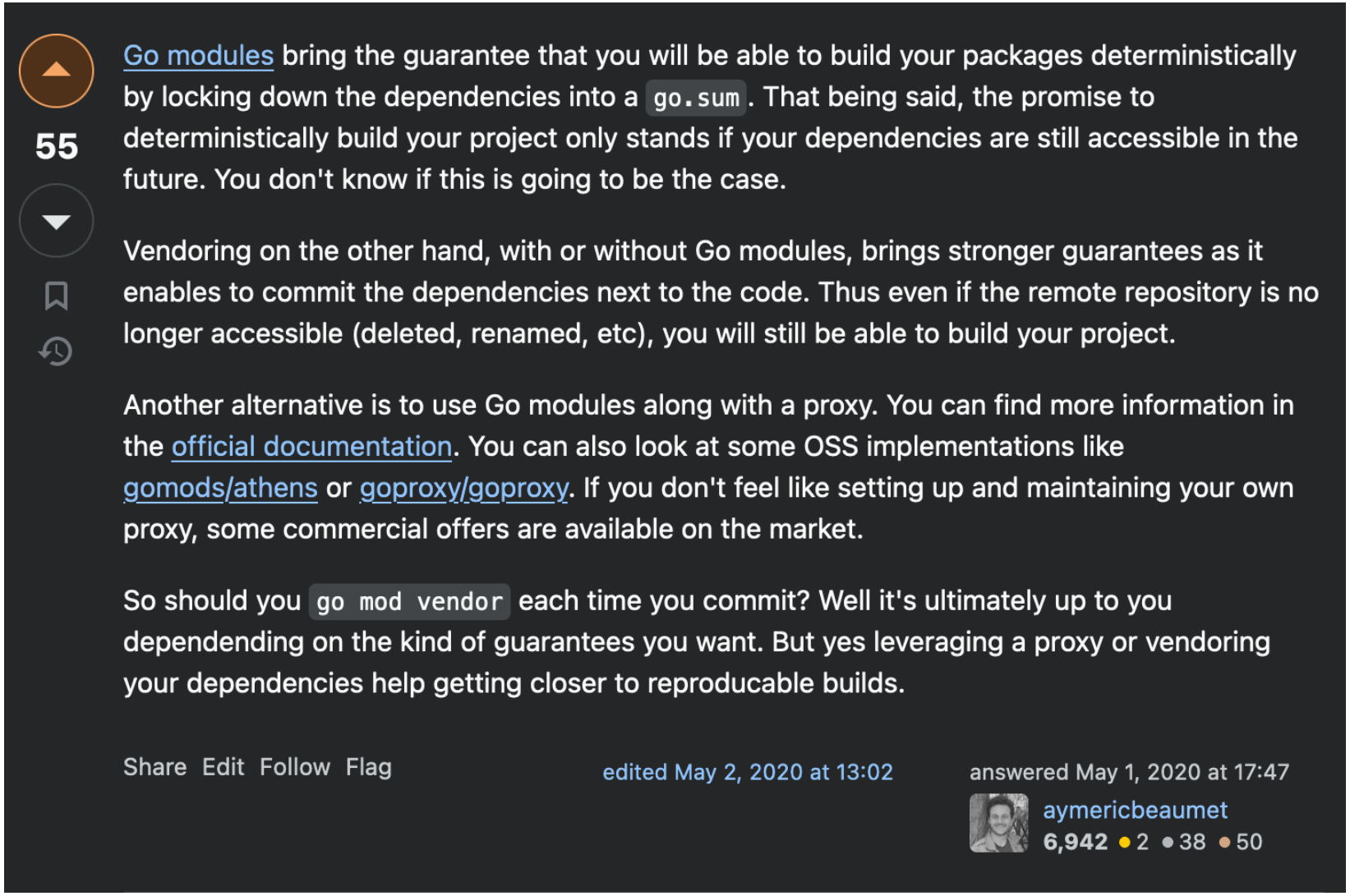
/pkg



Common directories

- /cmd
- /internal
- /pkg
- /vendor
  - Contains dependencies for the project

A note on vendor-ing



55

Go modules bring the guarantee that you will be able to build your packages deterministically by locking down the dependencies into a `go.sum`. That being said, the promise to deterministically build your project only stands if your dependencies are still accessible in the future. You don't know if this is going to be the case.

Vendoring on the other hand, with or without Go modules, brings stronger guarantees as it enables to commit the dependencies next to the code. Thus even if the remote repository is no longer accessible (deleted, renamed, etc), you will still be able to build your project.

Another alternative is to use Go modules along with a proxy. You can find more information in the official documentation. You can also look at some OSS implementations like gomods/athens or goproxy/goproxy. If you don't feel like setting up and maintaining your own proxy, some commercial offers are available on the market.

So should you `go mod vendor` each time you commit? Well it's ultimately up to you dependending on the kind of guarantees you want. But yes leveraging a proxy or vendoring your dependencies help getting closer to reproducable builds.

Share Edit Follow Flag

edited May 2, 2020 at 13:02

answered May 1, 2020 at 17:47

aymericbeaumet

6,942 ● 2 ● 38 ● 50

https://stackoverflow.com/a/61547800

Other directories

- /api – OpenAPI/Swagger, protobufs
- /docs – design docs, architectures
- /examples – examples for apps/libraries
- /hack – Dockerfiles, Makefiles, shell scripts
- /tests – e2e tests, integration tests
- /web – static web assets, server-side templates

Thank you!

@shubham1172