Standardizing Errors in Go: A Practical Guide with Dapr

Cassie Coyle

Table of Contents

Speaker & Team

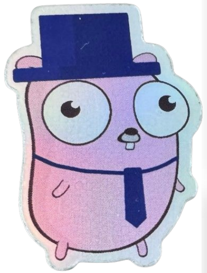
Cassie Coyle
Software Engineer
https://cassiecoyle.me/
Github - cicoyle
Twitter - cassie1coyle

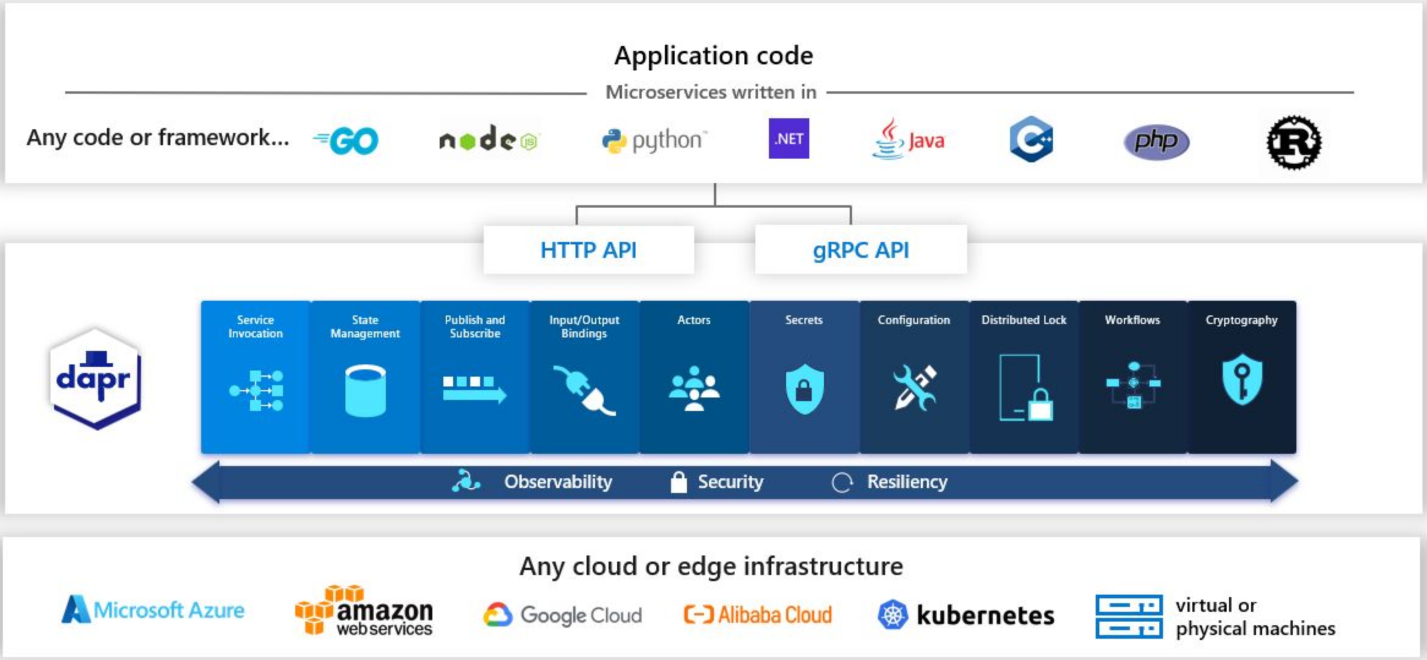
The Dapr OSS Team

Diagrid

Introduction

Dapr




Application code
Microservices written in
Any code or framework...
GO
node
python
.NET
Java
php
HTTP API
gRPC API
dapr
Service Invocation
State Management
Publish and Subscribe
Input/Output Bindings
Actors
Secrets
Configuration
Distributed Lock
Workflows
Cryptography
Observability
Security
Resiliency
Any cloud or edge infrastructure
Microsoft Azure
amazon webservices
Google Cloud
Alibaba Cloud
kubernetes
virtual or physical machines

Dapr Community Feedback

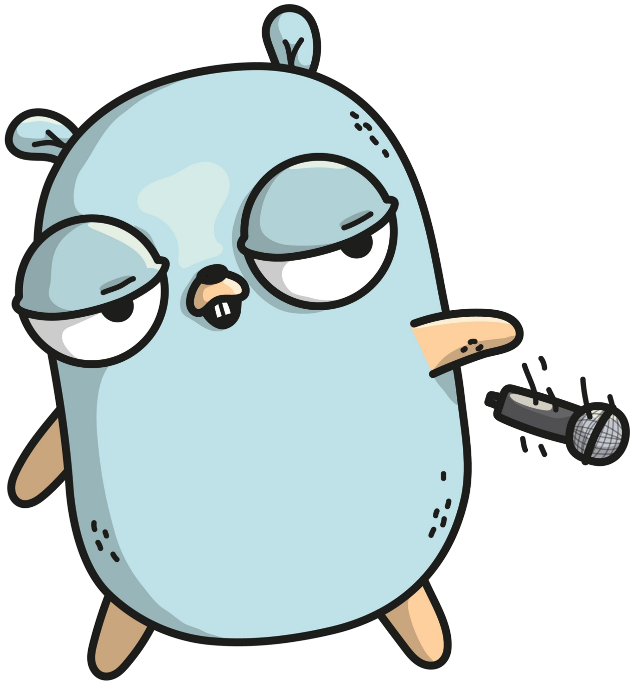
Frankly, our whole team loves Dapr. This is the backbone of our services.

DEVELOPER, LARGE SERVICES CO.

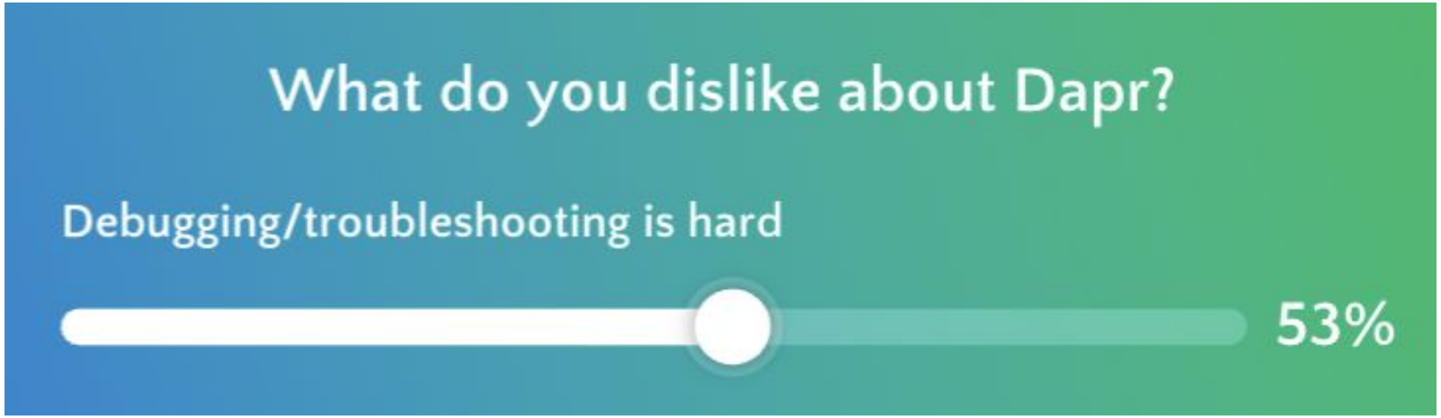
Dapr doesn't have a lot of expense at the beginning, and there is a lot of payback.

DEVOPS ENGR. LARGE FINANCIAL SERVICES CO.

Dapr Community Feedback



Dapr Community Feedback



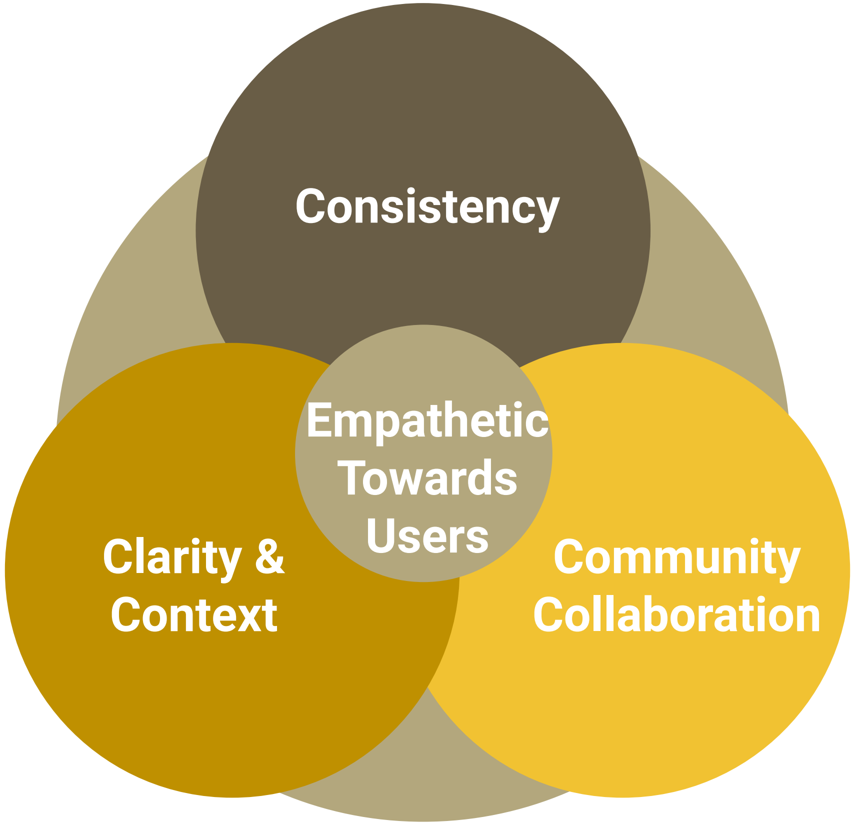
What do you dislike about Dapr?

Debugging/troubleshooting is hard

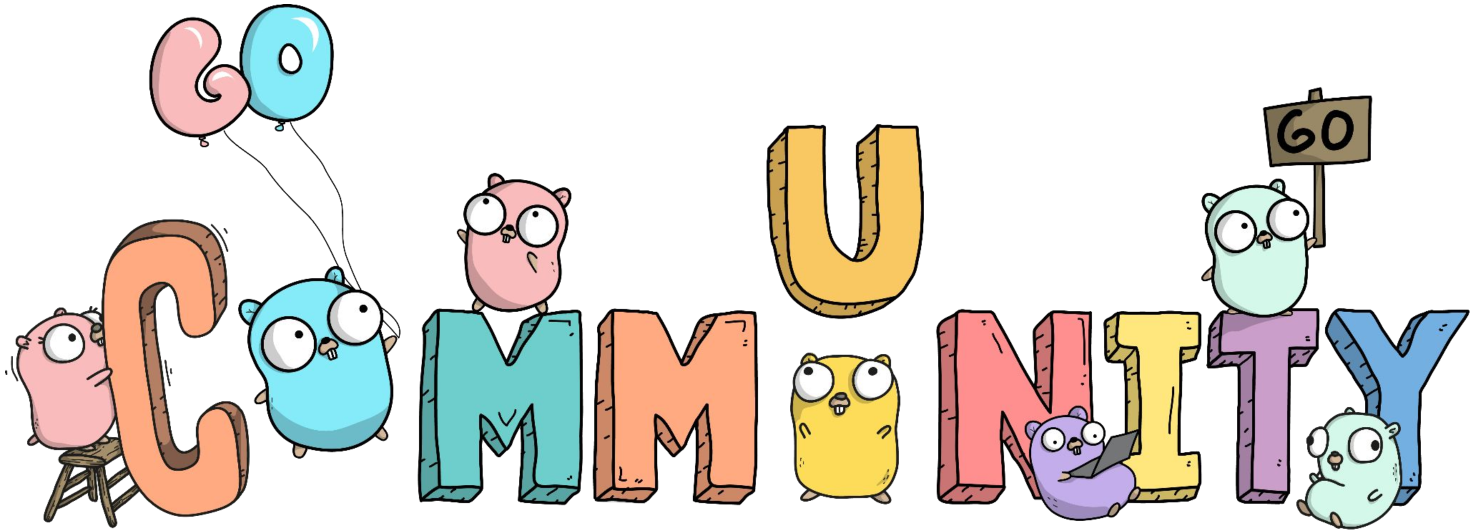
53%

Importance of Error Standardization

The Importance of Error Standardization

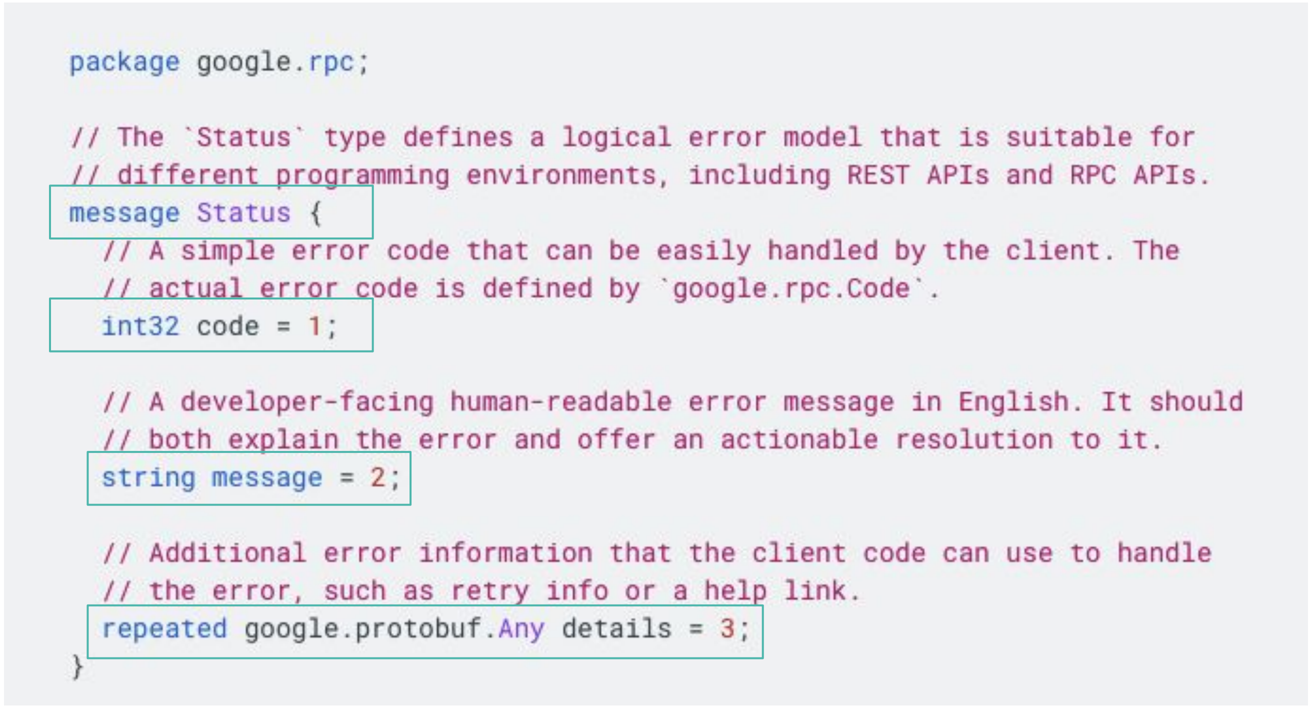


The Importance of Error Standardization



Richer Error Model

Richer Error Model



```
package google.rpc;

// The `Status` type defines a logical error model that is suitable for
// different programming environments, including REST APIs and RPC APIs.
message Status {
  // A simple error code that can be easily handled by the client. The
  // actual error code is defined by `google.rpc.Code`.
  int32 code = 1;

  // A developer-facing human-readable error message in English. It should
  // both explain the error and offer an actionable resolution to it.
  string message = 2;

  // Additional error information that the client code can use to handle
  // the error, such as retry info or a help link.
  repeated google.protobuf.Any details = 3;
}
```

Error Details

- ErrorInfo (required)
- RetryInfo
- DebugInfo
- QuotaFailure
- PreconditionFailure
- BadRequest
- RequestInfo
- ResourceInfo
- Help
- LocalizedMessage
- QuotaFailure_Violation
- PreconditionFailure_Violation
- BadRequest_FieldViolation
- Help_Link

Error Detail

- **ErrorInfo** (required)

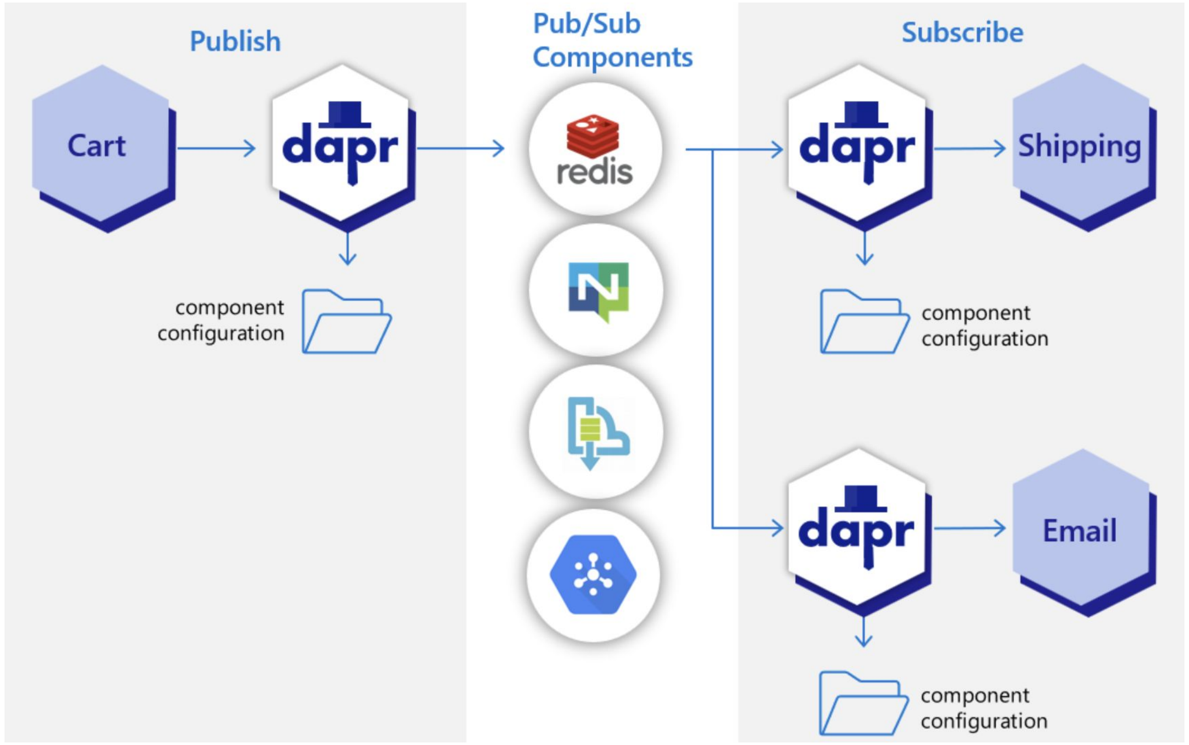
Error Detail

- **ErrorInfo** (required) $\Rightarrow$

ErrorInfo:
- Domain: dapr.io
- Reason: <reason>
- Metadata: <metadata>

Enriched Errors

PubSub API Overview



Enriched Error for PubSub API

```
$ grpcurl -H 'dapr-app-id: sub' -d
'{"pubsub_name":"fakeKafka",
"topic":"fakeTopic"}' -plaintext
localhost:57534
dapr.proto.runtime.v1.Dapr.PublishEvent
```

```
ERROR:
Code: InvalidArgument
Message: pubsub fakeKafka is not found
Details:
1)      {
"@type":
"type.googleapis.com/google.rpc.ErrorInfo",
"domain": "dapr.io",
"reason": "DAPR_PUBSUB_NOT_FOUND",
}
2)      {
"@type":
"type.googleapis.com/google.rpc.ResourceInfo",

"description": "pubsub fakeKafka is not found",
"resourceName": "fakeKafka",
"resourceType": "pubsub"
}
```

Expanded Enrichment

Details:
1) {
"@type": "type.googleapis.com/google.rpc.**HelpLink**",
"url":
"https://docs.dapr.io/developing-applications/building-blocks/pubsub/pubsub-overview",
"description": "Dapr docs pubsub overview",
}

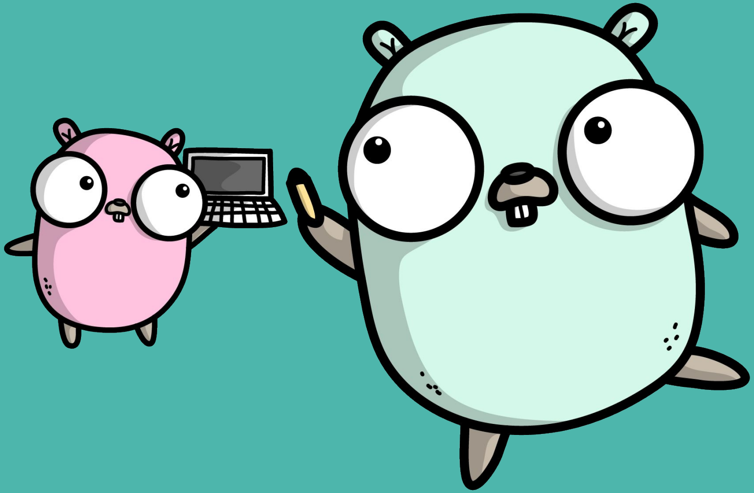
Consumption of the Errors pkg in dapr/kit repo

Consumption of the Errors pkg

```
err := errors.NewBuilder(grpcCode, httpCode, msg, tag)

return err.WithErrorInfo(
       errors.CodePrefixPubSub+errCode,
      p.metadata,
).Build()
```

Implementation Details



Implementation Details of the Errors pkg

```
type Error struct {
	details []proto.Message
	grpcCode grpcCodes.Code
	httpCode int
	message string
	tag string
}
```

```
type ErrorBuilder struct {
	err Error
}
```

Implementation Details of the Errors pkg

```
func NewBuilder(grpcCode grpcCodes.Code, httpCode int, message string,
tag string) *ErrorBuilder {
	return &ErrorBuilder{
		err: Error{
			details:  make([]proto.Message, 0),
			grpcCode: grpcCode,
			httpCode: httpCode,
			message:  message,
			tag:      tag,
		},
	}
}
```

Implementation Details of the Errors pkg

```
func (b *ErrorBuilder) WithErrorInfo(reason string, metadata
map[string]string) *ErrorBuilder {
	errorInfo := &errdetails.ErrorInfo{
		Domain:   Domain,
		Reason:   reason,
		Metadata: metadata,
	}
	b.err.details = append(b.err.details, errorInfo)

	return b
}
```

Implementation Details of the Errors pkg

```
func (b *ErrorBuilder) Build() error {
	// Check for required ErrorInfo
	containsErrorInfo := false
	for _, detail := range b.err.details {
		if _, ok := detail.(*errdetails.ErrorInfo); ok {
			containsErrorInfo = true
			break
		}
	}
	if !containsErrorInfo {...}

	return b.err
}
```

Implementation Details of the Errors pkg

```
func (b *ErrorBuilder) Build() error {
        // Check for required ErrorInfo
        containsErrorInfo := false
        for _, detail := range b.err.details {
                if _, ok := detail.(*errdetails.ErrorInfo); ok {
                        containsErrorInfo = true
                        break
                }
        }
        if !containsErrorInfo {...}

        return b.err
}
```

How To: Build the Future with us

Build the Future with us

- Errors pkg in dapr/kit repo
- Get involved in Dapr:
  - https://github.com/dapr/dapr/issues
  - https://github.com/dapr/kit/tree/main/errors