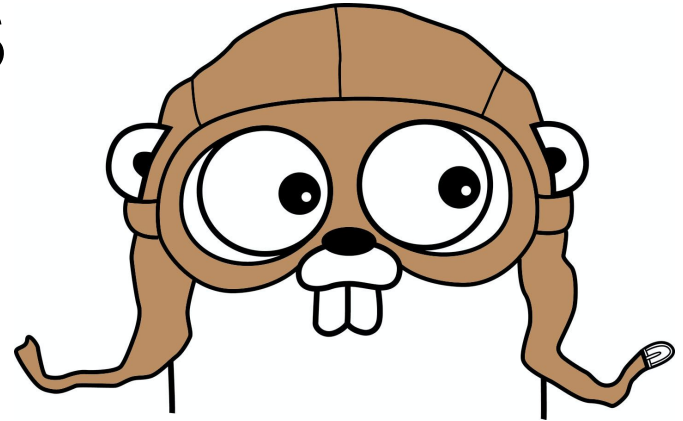# Gophers guide to Data Streaming at Scale with Benthos

benthos.dev

Mihai Todor
23.11.2021

Golang North East Meetup

Disclaimers

➤ Thoughts expressed here are my own

➤ Benthos is owned and maintained by Ash:
https://twitter.com/Jeffail/

➤ Ash designed and built Benthos from scratch
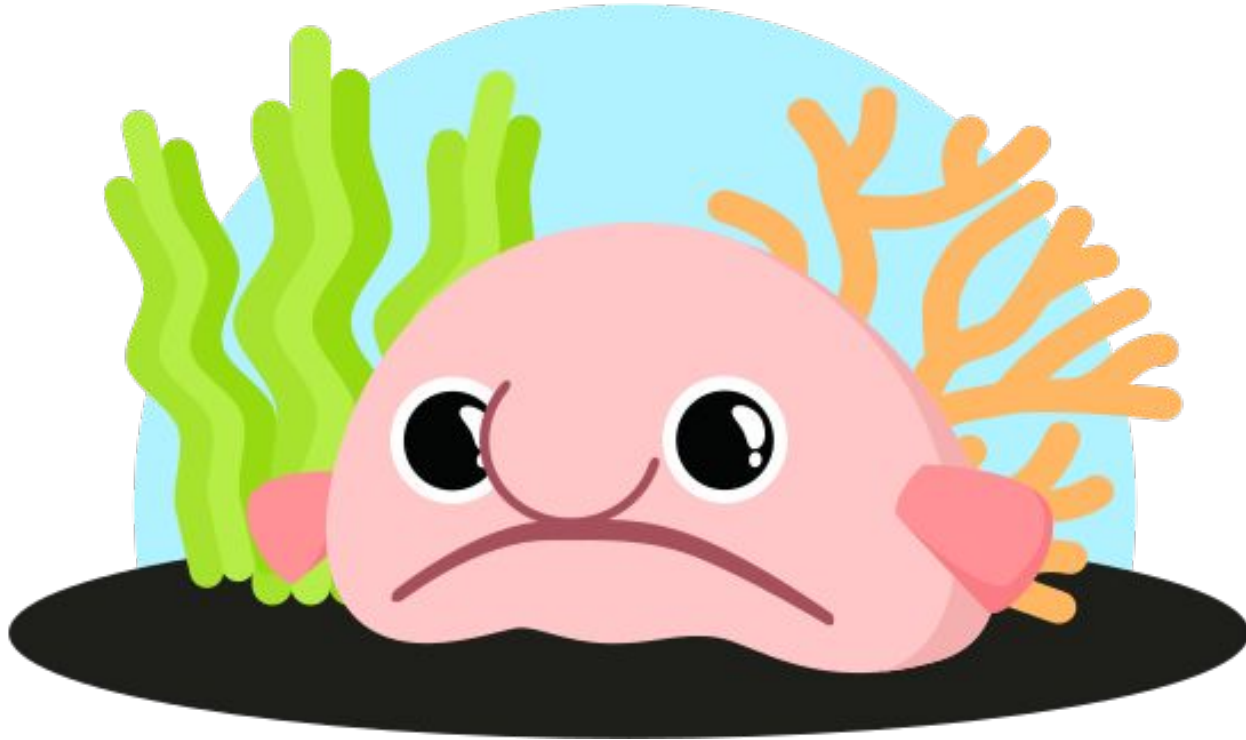
➤ I help out and send PRs occasionally

Ashley Jeffs
@Jeffail Follows you

A bland person, building benthos.dev.
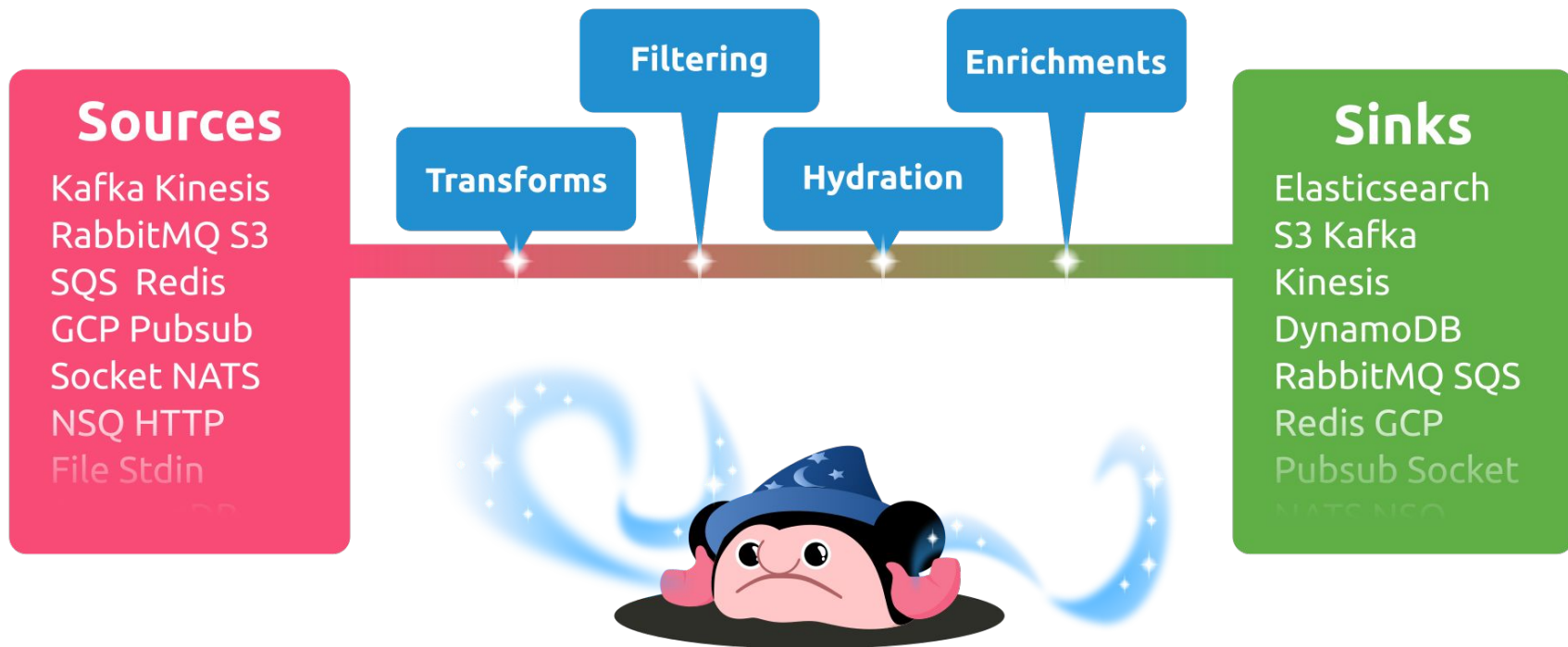
Github: github.com/Jeffail/benthos
YouTube: youtube.com/c/Jeffail

What is Benthos? https://www.benthos.dev/

Hint: It has nothing to do with deep sea fish, except for the logo

What is Benthos for?

"Fancy stream processing made operationally mundane" - Ash Jeffs

What is Data Streaming?

➢ Go Time Podcast episode #192:
  https://changelog.com/gotime/192

➢ Realtime ingestion and
  transformation of (small) data
  records from one or multiple
  sources in parallel

➢ Batch processing of large volumes
  of data records

➢ Event sourcing blurry lines

Data engineering

➢ The Data Stack Show Podcast episode #60: https://datastackshow.com/podcast/architecting-a-boring-stream-processing-tool-with-ashley-jeffs-of-benthos/

➢ Project history

➢ Origins of the awesome logo

Boringly easy to use

Install

curl -Lsf https://sh.benthos.dev | bash

Make a config

benthos create nats/protobuf/aws_sqs > ./config.yaml
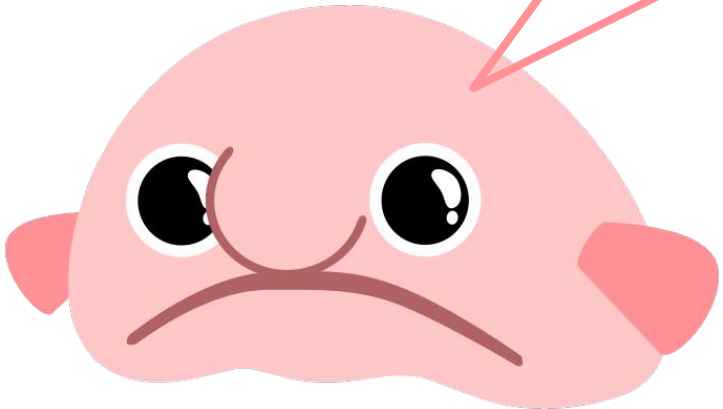
Run

benthos -c ./config.yaml

Features

- ➢ Declarative YAML-based configuration

- ➢ Single message transforms

- ➢ Stateless

- ➢ At least once delivery

- ➢ Metrics and logging

- ➢ Custom Plugins

- ➢ Written in Go

Bloblang

Custom DSL for arbitrary data transforms
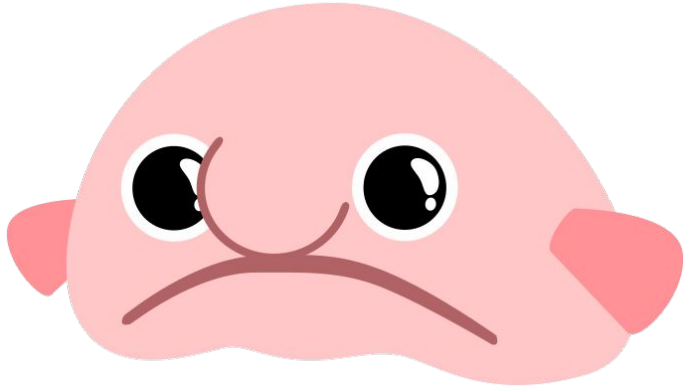
```
root.new_doc = match this.doc {
  this.type == "article" => this.article
  this.type == "comment" => this.comment
  _ => this
}
```

```
{
 "doc": {
   "type": "article",
   "article": {
     "id": "foo",
     "content": "qux"
   }
 }
}
```

```
{
 "new_doc": {
   "id": "foo",
   "content": "qux"
 }
}
```

Deployment models

Standalone CLI app

Serverless

Kubernetes

Importing Benthos as a library

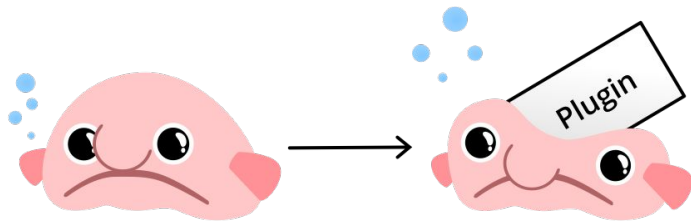> go get github.com/Jeffail/benthos/v3/public/components/all

```go
package main

import (
    "context"

    "github.com/Jeffail/benthos/v3/public/service"

    // Import all standard Benthos components
    _ "github.com/Jeffail/benthos/v3/public/components/all"
)

func main() {
    service.RunCLI(context.Background())
}
```

Writing a custom Benthos plugin

```go
type processor struct{}

func (r *processor) Process(ctx context.Context, m *service.Message) (service.MessageBatch, error) {
    println("foobar")
    return nil, nil
}


func (r *processor) Close(ctx context.Context) error { return nil }


func init() {
    _ = service.RegisterProcessor("foobar",
        service.NewConfigSpec(),
        func(conf *service.ParsedConfig, mgr *service.Resources) (service.Processor, error) {
            return &processor{}, nil
        },
    )
}
// ./plugin create stdin/foobar/stdout > config.yaml
// ./plugin -c config.yaml
```

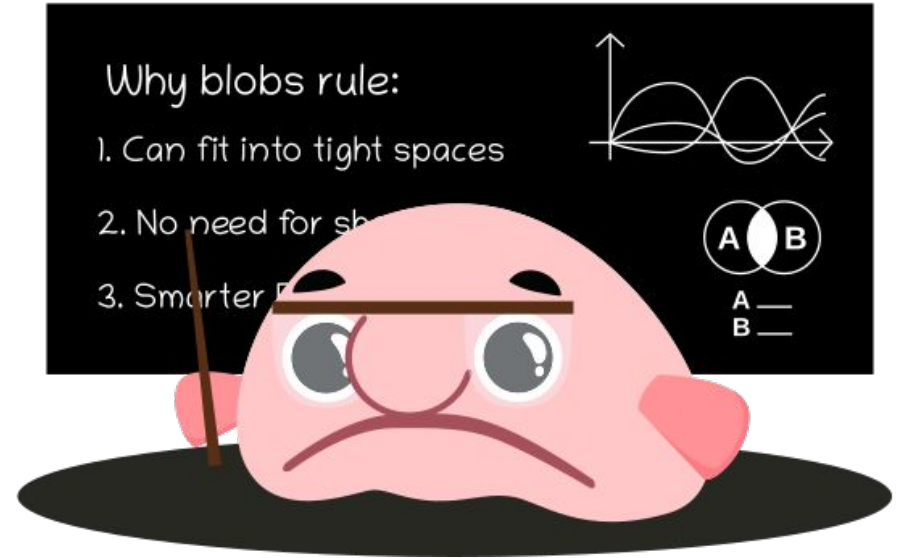# Future enhancements

More adapters

Custom plugins

Project S

Why Open Source?

➢ Built on the shoulders of giants

➢ Community-driven features, enhancements and bug fixes

➢ High quality standards enforced uniformly

➢ Open issue tracker and permanent change history

➢ Avoids vendor-driven lock-in

Open Source Needs You!

Community https://www.benthos.dev/community

Discord

https://discord.gg/6VaWjzP

slack

https://invite.slack.golangbridge.org
#benthos channel

Thank you!

➢ https://www.linkedin.com/in/mtodor/

➢ https://twitter.com/MihaiTodor

➢ https://github.com/mihaitodor