



London Gophers

19/02/2025





Hi



<https://ainsley.dev>



[@ainsleydev](https://twitter.com/ainsleydev)



linkedin.com/in/ainsleyclark



github.com/ainsleyclark







How Just Eat uses tooling to deploy Go microservices in minutes



JET Connect

Scaling Up

Demo

**Going
Further**

Results



JET Connect



What's JET Connect?

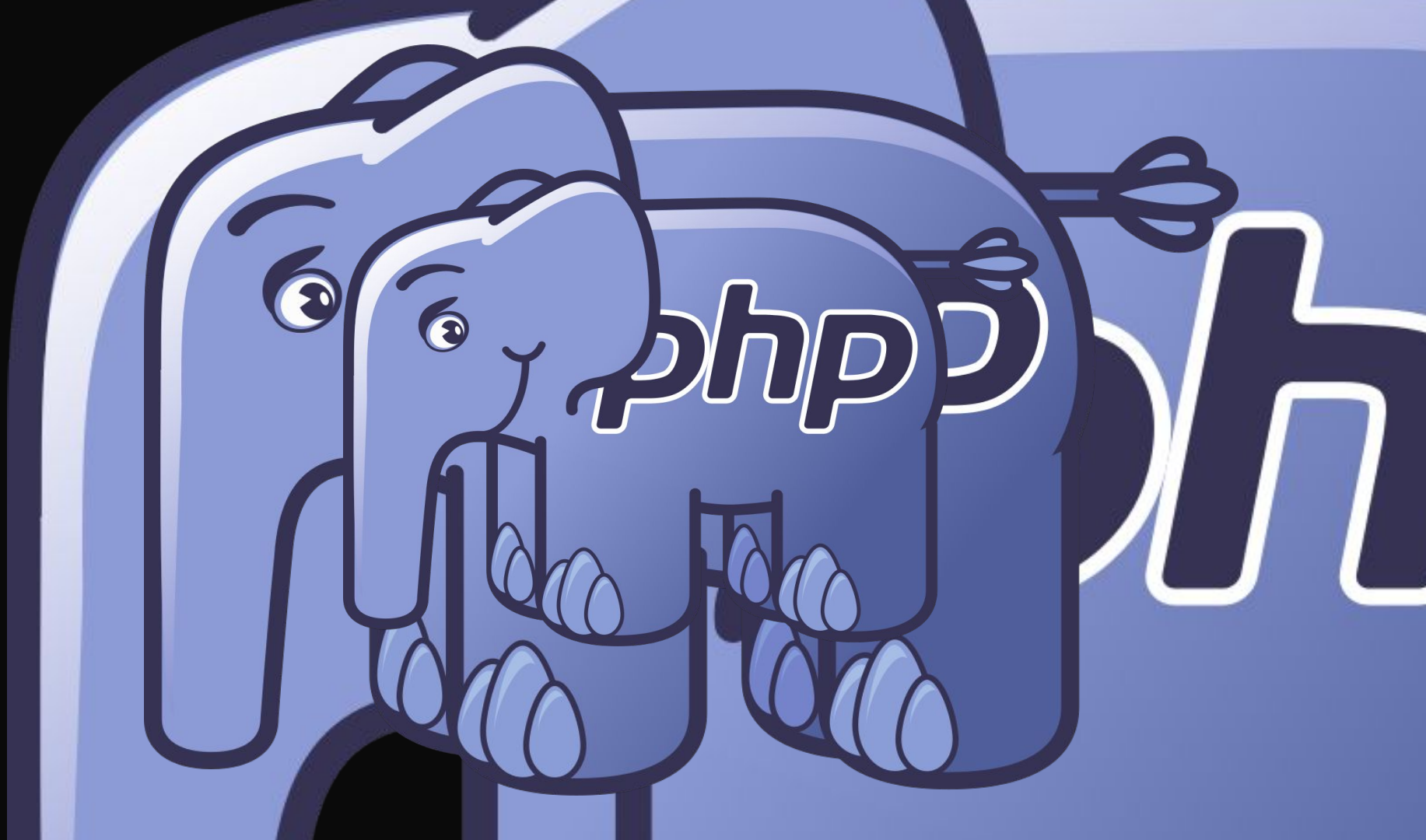
- Founded in 2013 as an integration platform.
- Unified order and menu processing.
- Partners are able to use one unified system.







Scaling Up





go-kit

```
package ordering;
```

```
service OrderSender {  
    rpc SendOrder(OrderRequest) returns (OrderResponse);  
}
```



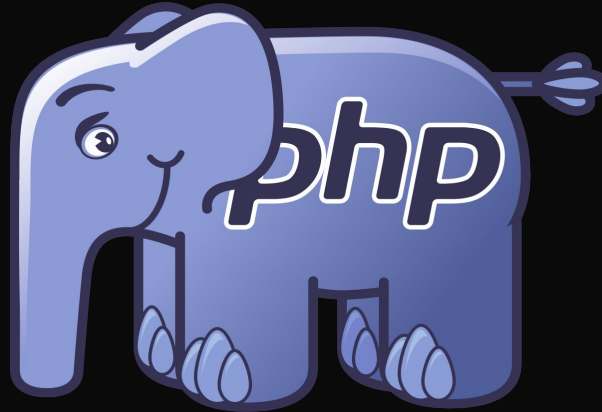
Dominos



KFC



McDonald's



Monolith



Subway



Problem Areas

- Maintenance
- Updates
- No E2E testing
- Needed a way to easily create & update services.





Switching To Go

- Needed something that scaled.
- So we switched to Go.
- But needed to come up with a plan and prerequisites for a tool.





Prerequisites

- Generate and update a service in seconds.
- Define HTTP APIs with industry standard docs.
- Subscribe to events with as little code as possible.
- Abstract as much of the infra as possible.
- All CI/CD auto generated so rolling out was easy.
- All changes to main deployed to production, safely (E2E).
- Developers can run services locally with ease.
- Patterns that are used frequently abstracted.

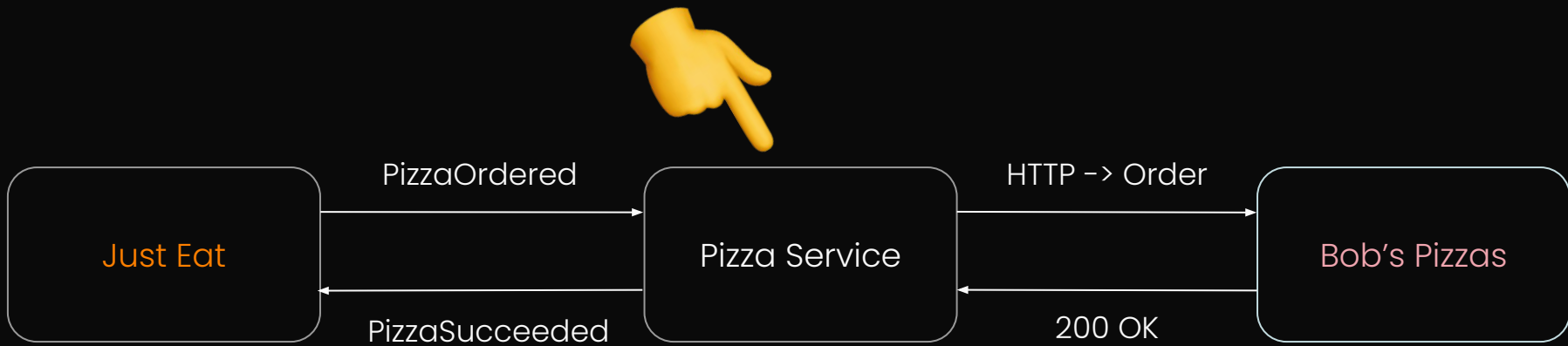


GoKit

Microservice Development Tooling



Demo





go-kit.sh

```
go-kit new pizza-service
```



internal/app/service.go

What APIs will your service produce or consume?

☐ HTTPS

☒ Events: Consume events

☒ Events: Produce events

☐ Database

☐ Object Storage

> ☒ HTTP Client



api
cmd/app (main.go)
internal
pkg
resources/helm
.gitignore
.golangci.yml
Dockerfile
README.md
go.mod
go.sum
service.json



internal/app/service.go



```
func RunService(  
    client *http.Client,  
    consumer eventbus.Consumer,  
    producer eventbus.Producer,  
    error {  
        return nil  
    }  
}
```



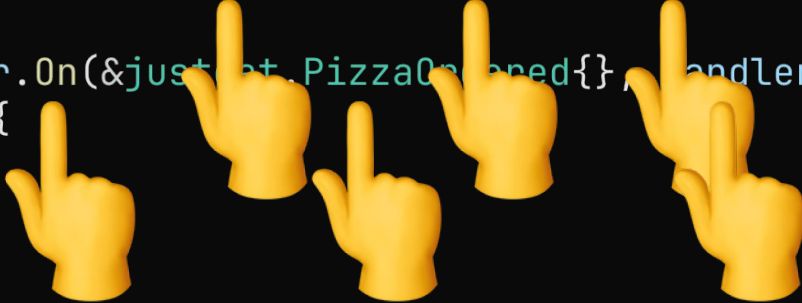
internal/handlers/pizza-ordered.go

```
func PizzaHandler(  
    client *http.Client,  
    producer eventbus.Producer,  
    ) eventbus.EventHandler {  
    return func(ctx context.Context, event any, headers ...eventbus.Header) error {  
        ev := event.(*justeat.PizzaOrdered)  
  
        // Tell Partner about Pizza...  
        order := Order{}  
        client.Post(http.MethodPost, "/order", bytes.NewBuffer(order.Bytes()))  
        if err != nil {  
            return err  
        }  
  
        producer.Emit(ctx, &justeat.PizzaSucceeded{  
            RestaurantID: "1234",  
        })  
    }  
}
```




internal/app/service.go

```
func RunService(  
    client *http.Client,  
    consumer eventbus.Consumer,  
    producer eventbus.Producer,  
) error {  
    handler := handlers.PizzaHandler(client, producer)  
  
    err := consumer.On(&justEat.PizzaOrdered{}, handler)  
    if err != nil {  
        return err  
    }  
  
    return nil  
}
```





go-kit.sh

```
go-kit update
```



service.json

```
{
  "event_bus": {
    "consumes_topics": [
      {
        "name": "pizza-ordered",
        "go_name": "justeat.PizzaOrdered",
        "timeout": 30,
        "retry_count": 5,
        "worker_pool": 25,
        "replica_count": 3,
        "pod_resources": {
          "cpu": "small",
          "memory": "small"
        },
        "consumer_lag_alert": {
          "number_of_messages": 50,
          "time_period": "2m"
        }
      }
    ]
  }
}
```



Events

Consumes

Name	CPU	Memory	Worker Pool
justeat.PizzaOrdered	small 50m	small 50Mi	25

Produces

Name
justeat.PizzaSucceeded





Deployment

Pull Request



Add a title

Creating Pizza Service

Add a description

Write

Preview

H

B

I

≡

<>

🔗

≡

≡

≡

🔗

@

📄

↩

Context

<!-- Required: What is this PR achieving? Provide a brief description on a high level of the problem or feature being addressed in this PR. Include any relevant background information that helps explain why these changes are necessary. -->

Changes

<!-- Required: Describe the changes you have made in this PR. Focus on what has been added, removed, or modified. -->

Refactors

<!-- Describe any refactoring done as part of this PR. Explain the purpose and how it improves the codebase. -->



Markdown is supported



Paste, drop, or click to add files

Capability Testing



Drift Detection



```
service.json

consumer.OnAll(map[proto.Message]eventbus.EventHandler{
    &justeat.MyEvent{}: func(ctx context.Context, event any, headers ...eventbus.Header) error {
        return nil
    },
})
```



github-actions bot commented 6 minutes ago



Generated by go-kit ci: Service Validation

Consumer topic "my-event" in service.go is not a used topic in service.json

Consumer topics used in service.go

Rollout



Re-run triggered 2 days ago

Status

Total duration

Artifacts

ainsley-clark 6e037ad main

Success

6m 21s

1

release.yml

on: push

✓ create-ecr / create-ecr 5s

✓ ens... / create-sentry-project 1s

✓ test-unit / test-unit 2m 16s

✓ mono... / monorepo-tests 9m 16s

✓ tool_build-amd / build-amd 34s

✓ tool_build-a... / build-arm 2m 37s

✓ tool_build...

[]

-

+

Metrics



▼ Events Produced

Event Rates ⓘ



Event Emit Duration Per Event ⓘ



Event Emit Duration Overall ⓘ



▼ Events Consumed

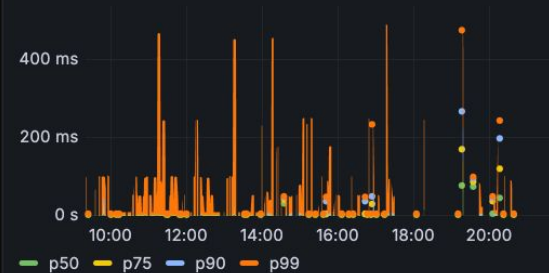
Event Rates ⓘ



Event Error Rates ⓘ



Event Processing Duration Overall ⓘ





Recap

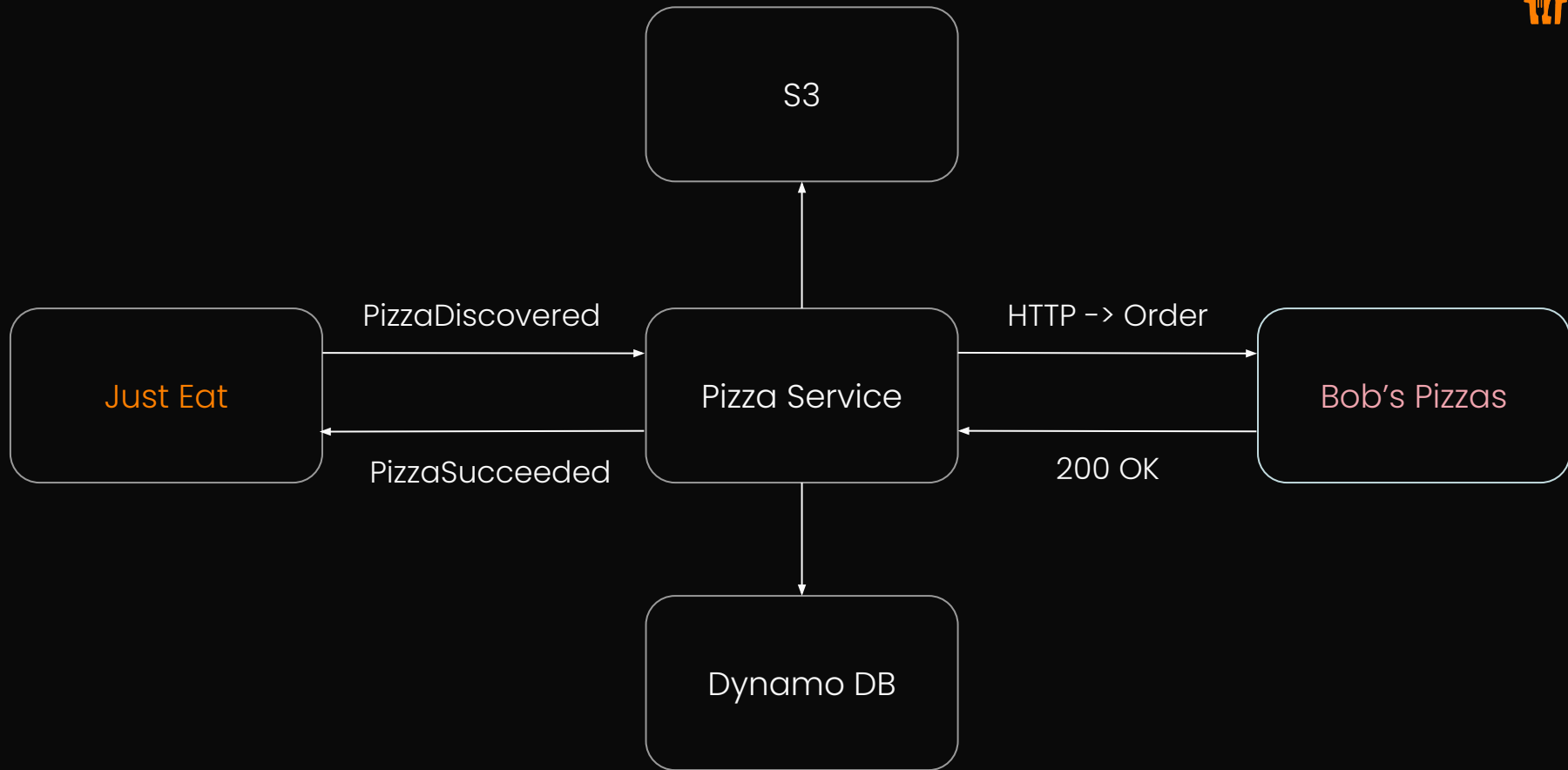
- Ran go-kit new
- Created one handler
- Attached it to a consumer
- And we've deployed a service with metrics, workflows and much more.





Going Further







service.json

```
{
  "resources": {
    "databases": [
      {
        "name": "analytics",
        "description": "Keeps track of basket amounts of orders",
        "attributes": null,
        "indexes": null
      }
    ],
    "object_stores": [
      {
        "name": "orders",
        "description": "Stores the order payload before injecting",
        "object_expire_days": 3
      }
    ]
  }
}
```



go-kit.sh

```
go-kit update
```



DynamoDB Tables

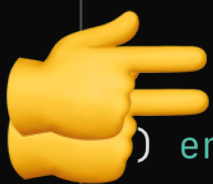
Name	Description	Composite
analytics	Keeps track of basket amounts of orders	✗

Stores

Name	Description	Type	🕒 Expiry
orders	Stores the order payload before injecting	object store	3 days



internal/app/service.go



```
func RunService(  
    client *http.Client,  
    consumer eventbus.Consumer,  
    producer eventbus.Producer,  
    db projections.ReadWriter,  
    store objstore.DownloadUploader,  
) error {  
    handler := handlers.PizzaHandler(client, producer, db, store)  
  
    err := consumer.On(&justeat.PizzaOrdered{}, handler)  
    if err != nil {  
        return err  
    }  
  
    return nil  
}
```





go-kit

```
type ReadWriter interface {  
    Reader  
    Writer  
}  
  
type Reader interface {  
    Read(ctx context.Context, id string, projection any) error  
}  
  
type Writer interface {  
    Write(ctx context.Context, data any, expires *time.Time) error  
}
```



internal/handlers/pizza-ordered.go

```
func PizzaHandler(
    client *http.Client,
    producer eventbus.Producer,
    db projections.ReadWriter,
    store objstore.DownloadUploader,
) eventbus.EventHandler {
    return func(ctx context.Context, event any, headers ...eventbus.Header) error {
        ev := event.(*justeat.PizzaOrdered)

        // Tell Partner Restaurant...
        order := Order{}
        client.Post(http.MethodPost, "/order", bytes.NewBuffer(order.Bytes()))

        err := db.Write(ctx, order.Analytics(), nil)
        if err != nil {
            return err
        }

        _, err := store.Upload(ctx, "orders", "/bobs-restaurant/" + time.Now().String(), order)
        if err != nil {
            return err
        }

        producer.Emit(ctx, &justeat.PizzaSucceeded{
            RestaurantID: "1234",
        })
    }
}
```



go-kit.sh

```
go-kit run
```



go-kit.sh



Building Service with "go"



Build Successful



Running App. You can stop it with ctrl + c.

Using in-memory for event bus

Using in-memory for projections

Using in-memory for object storage

DynamoDB available at: <http://localhost:8000>

MinIO available at: <http://localhost:9000>



Deploy...



Resource - DynamoDB - Analytics



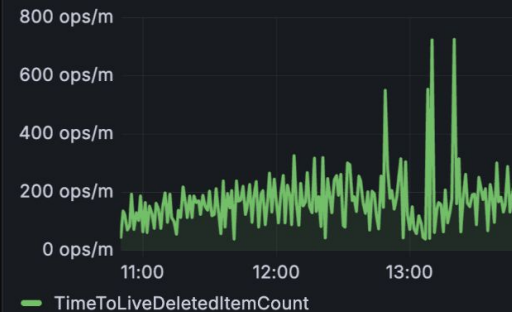
Consumed Capacity ⓘ



Successful Request Latency ⓘ



TTL Deletes



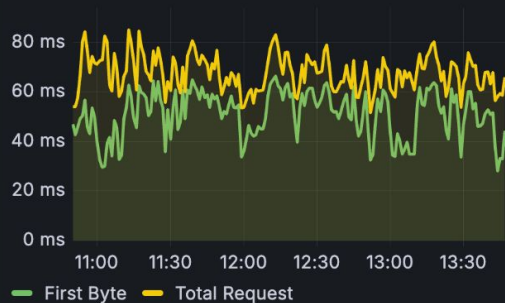
Resource - S3 - Orders



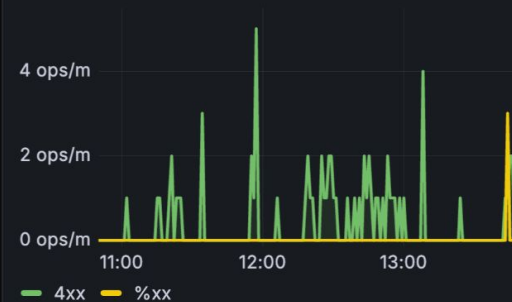
Operations Per Bucket ⓘ



Latency ⓘ



Errors Per Bucket ⓘ





NOT BAD



Results

Go Kit helps us to



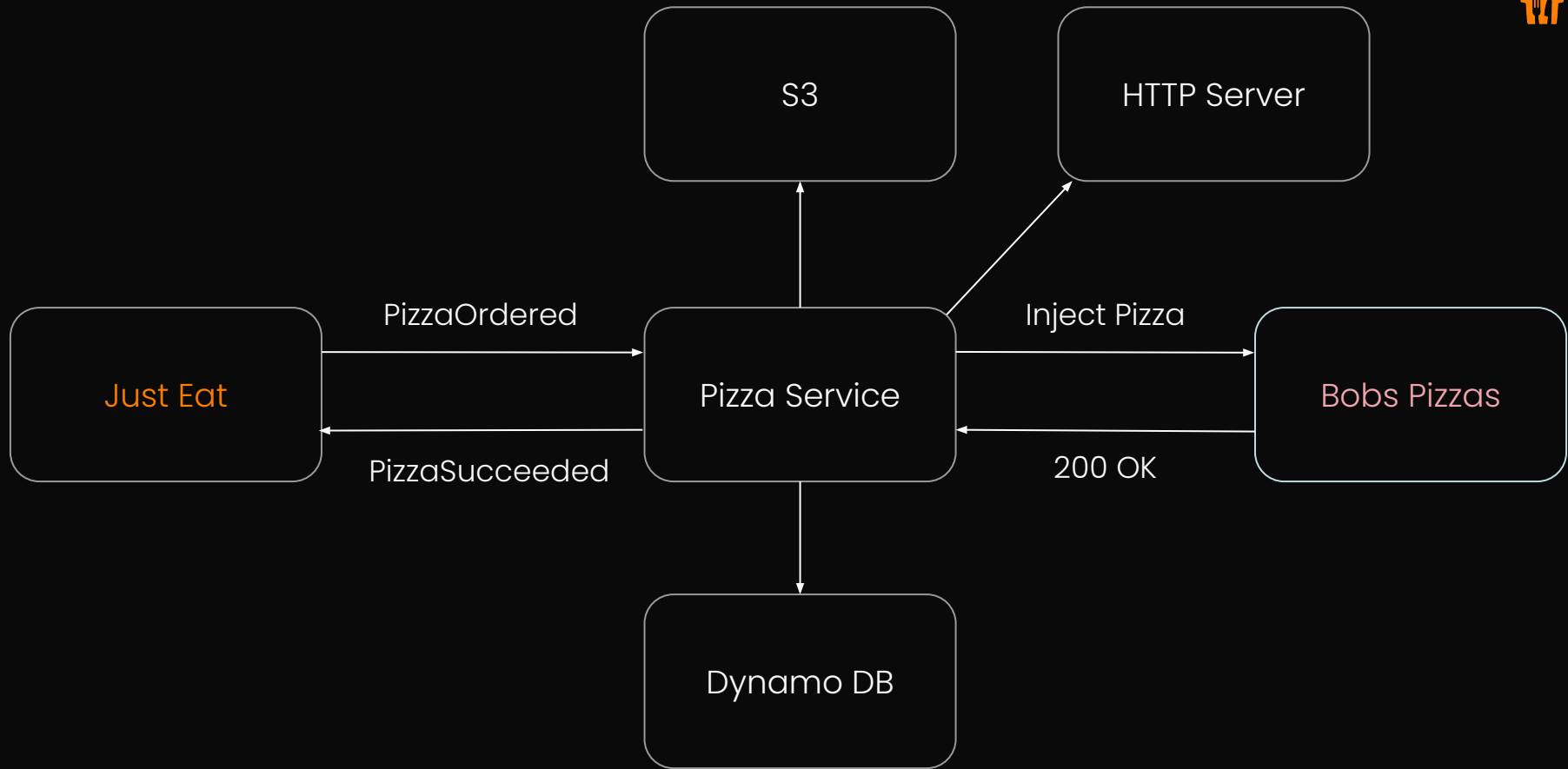
- Update our entire estate (150+ services) in one PR.
- Have naturally occurring SRE & updates over time.
- Create first class documentation without intervention
- Focus on business problems, not deployment.
- CI/CD updates and changes easily implemented.
- Helps us with patterns for file structure.
- Have extremely robust end to end testing.
- Scale to hundreds of millions of orders a month.

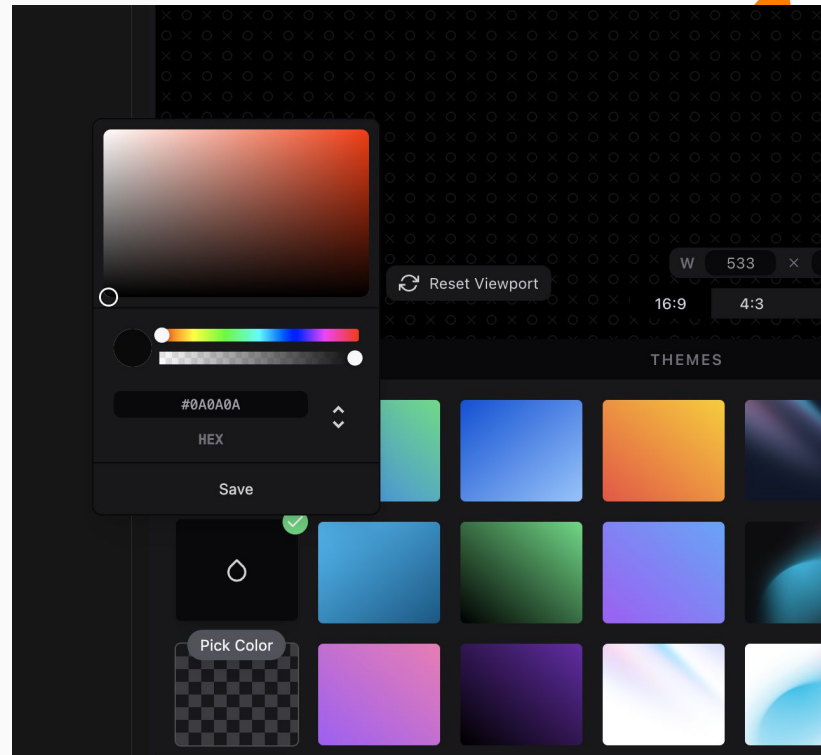
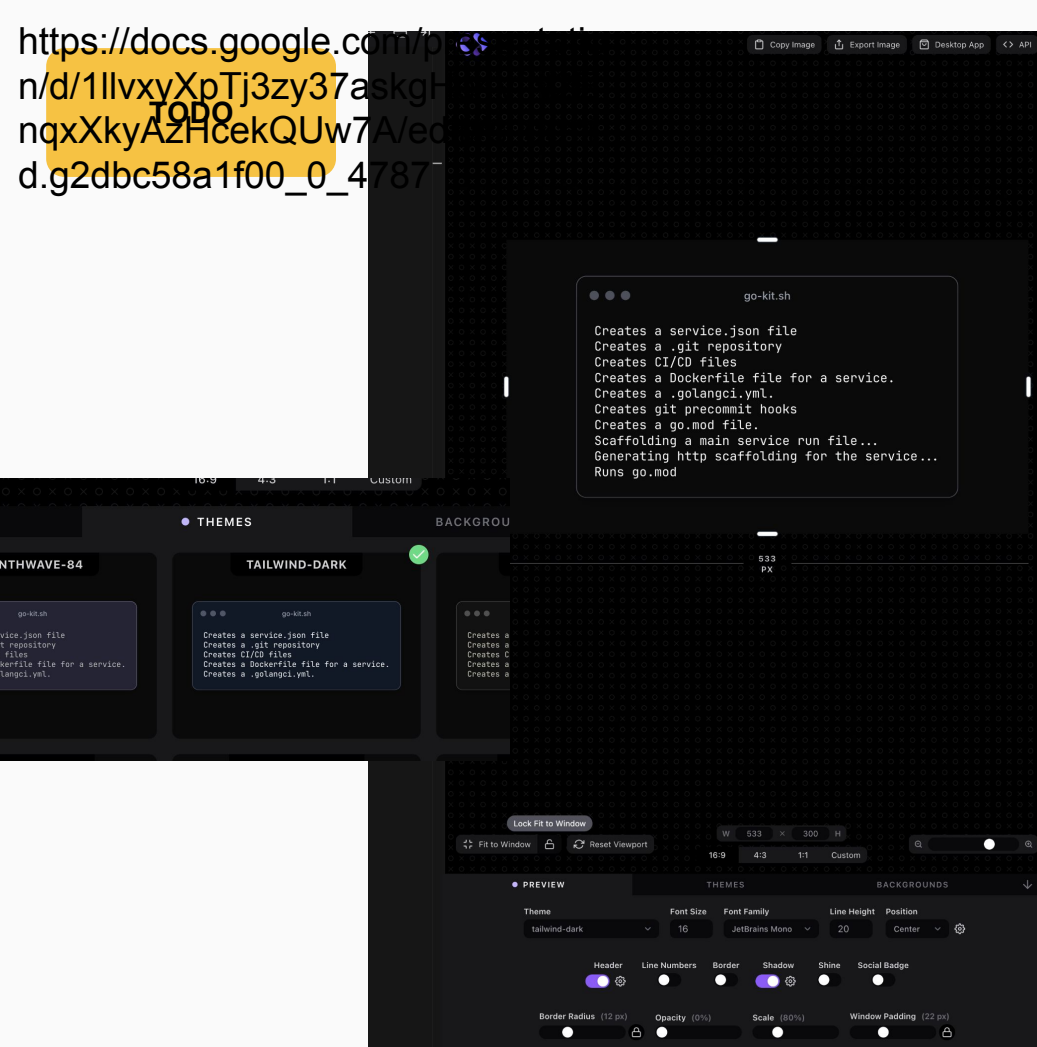


Thank you!

Questions?









<https://ainsley.dev>



[@ainsleydev](https://twitter.com/ainsleydev)



linkedin.com/in/ainsleyclark



github.com/ainsleyclark

