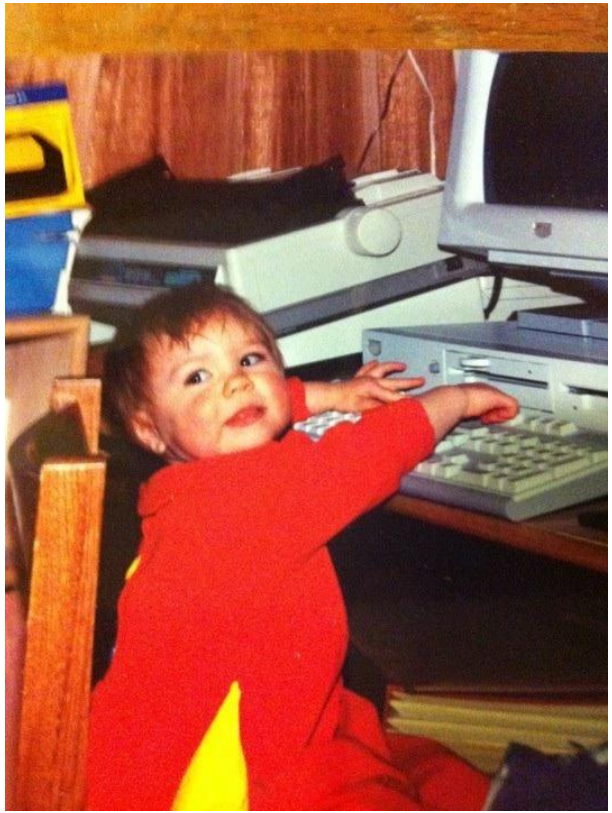
Prototyping a Go, GraphQL, gRPC microservice

Sadie Freeman
Senior Backend Engineer

GopherCon Chicago 2024

Me



Why build a Prototype/Proof of Concept?

"Does this thing work?"

"Can I use X with Z?"

"What is the level of effort?"

"We should use this because..."

Bonus: A teaching aid!

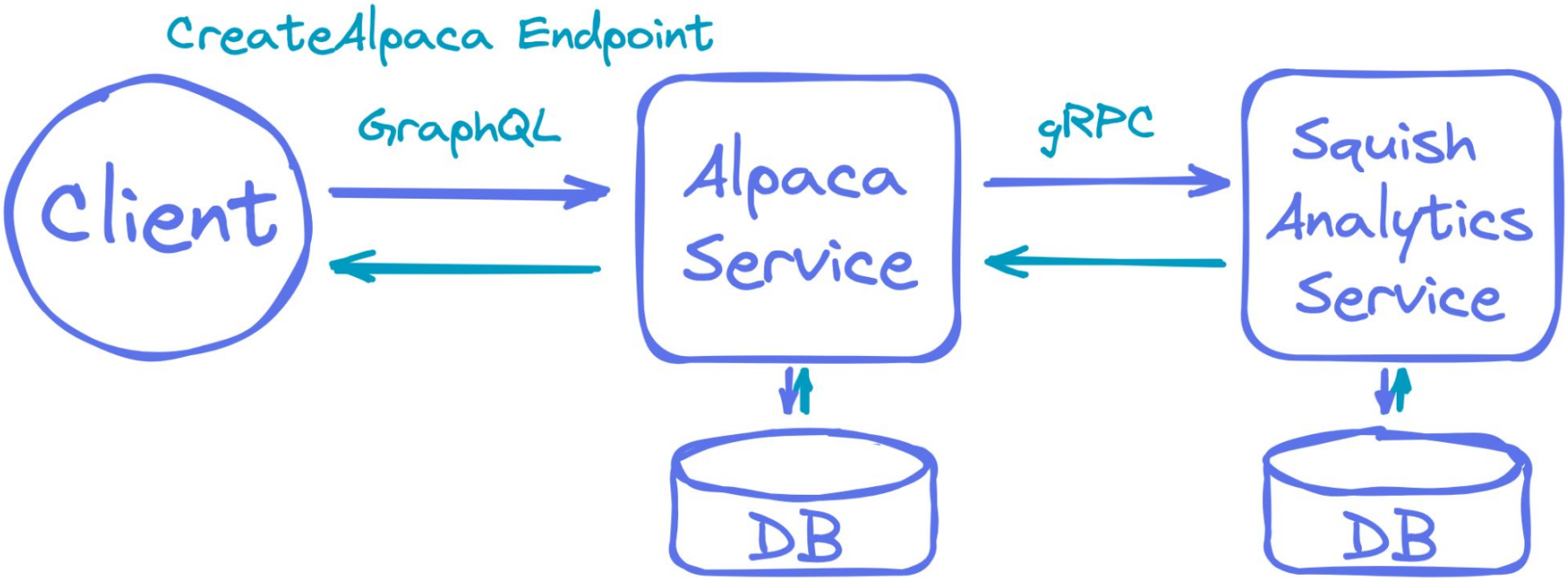
Prototype Essentials

- Document the process + pros/cons
- Use existing libraries/frameworks
- Use useful logs

- Overcomplicate
- Make assumptions about the target audience
- Build more than necessary

Fernando the Baby Alpaca

Architecture Overview



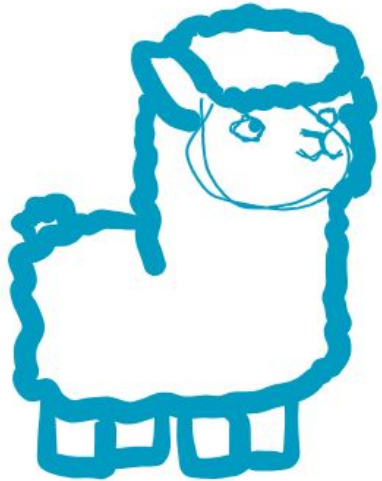
Go

- Fast/Performant
- Concurrent friendly
- Low barrier to entry
- Adoption is steadily increasing

Scaffolding a Go API

CreateAlpaca

```
Input: {
  "Name": "Fernando"
}

Output: {
  "ID": 1234,
  "Name": "Fernando"
}
```



Most basic Go server

```go
type Alpaca struct {
    Name string `json:"name"`
}

func main() {
    http.HandleFunc("/alpaca/", CreateAlpaca)
    log.Printf("Alpaca server up and running")
    http.ListenAndServe(":8080", nil)
}

func CreateAlpaca(w http.ResponseWriter, r *http.Request) {
    alpaca := Alpaca{}
    err := json.NewDecoder(r.Body).Decode(&alpaca)
    if err != nil {
        log.Printf("Bad Request: %v", err)
    }
    fmt.Printf("Alpaca name is %v", alpaca.Name)
}
```

GraphQL

- More flexible querying
- Front-End development friendly
- Fewer server requests
- Powerful pagination

Alpaca Service

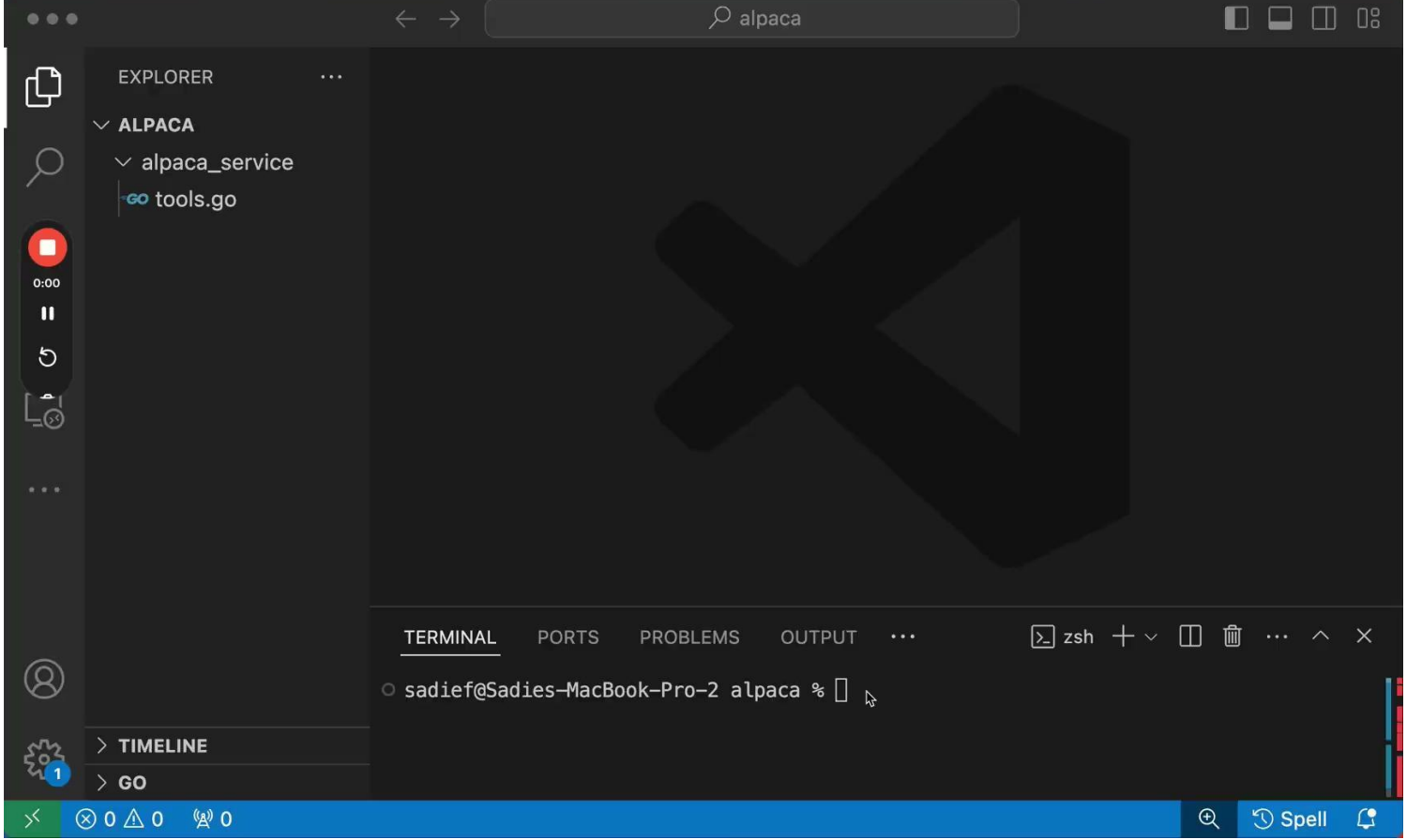
gqlgen library

```
`go get github.com/99designs/gqlgen`
`go run github.com/99designs/gqlgen init`
Update GraphQL Schema
Update schema.resolvers.go
`go run github.com/99designs/gqlgen generate`
`go run server.go`
Test CURL request
```

```
type Alpaca {
  id: ID!
  name: String!
}

input NewAlpaca {
  name: String!
  faceSquishRating: Int!
}

type Query {
  alpacas: [Alpaca!]!
}

type Mutation {
  createAlpaca(input: NewAlpaca!): Alpaca!
}
```

Alpaca Service

```go
// CreateAlpaca is the resolver for the createAlpaca field.
func (r *mutationResolver) CreateAlpaca(ctx context.Context, input model.NewAlpaca)
(*model.Alpaca, error) {
    log.Printf("Alpaca Name: %v | FaceSquishRating: %v", input.Name, input.FaceSquishRating)

    newAlpaca := &model.Alpaca{
        ID:   uuid.New().String(),
        Name: input.Name,
    }

    //TODO: Send FaceSquishRating to Squish Analytics Service
    return newAlpaca, nil
}
```


alpaca
EXPLORER
ALPACA
alpaca_service
tools.go
0:00
TERMINAL
PORTS
PROBLEMS
OUTPUT
zsh
sadief@Sadies-MacBook-Pro-2 alpaca %
TIMELINE
GO
0 0 0
Spell

gRPC

- Fast and efficient
- HTTP/2
- Protocol Buffers
- Clear Proto schema

Squish Analytics Service

Buf

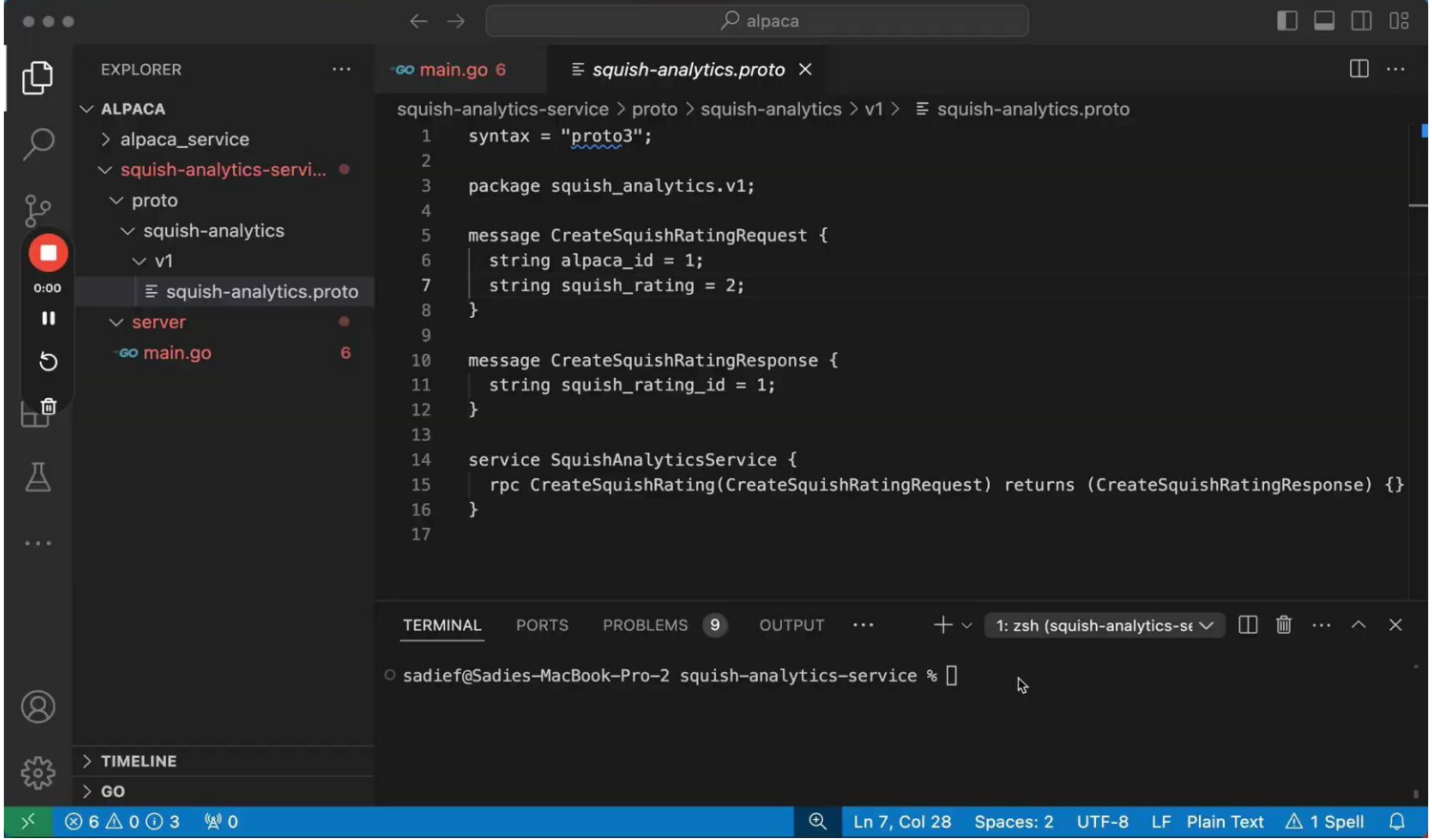
```
buf config init
buf build
buf generate
```

```
> gen
> proto
  buf.gen.yaml
  buf.yaml
  go.mod
  go.sum
```

```
syntax = "proto3";

package squish_analytics.v1;

message CreateSquishRatingRequest {
  string alpaca_id = 1;
  string squish_rating = 2;
}

message CreateSquishRatingResponse {
  string squish_rating_id = 1;
}

service SquishAnalyticsService {
  rpc CreateSquishRating(CreateSquishRatingRequest)
returns (CreateSquishRatingResponse) {}
}
```

Squish Analytics Service

```go
// CreateSquishRating adds a new rating entry for an Alpaca
func (s *squishAnalyticsServiceServer) CreateSquishRating(
    ctx context.Context,
    req *connect.Request[squish_analytics_v1.CreateSquishRatingRequest],
) (*connect.Response[squish_analytics_v1.CreateSquishRatingResponse], error) {
    squishRating := req.Msg.GetSquishRating()
    log.Printf("Received request for new Squish Rating: %v", squishRating)

    newRating := &squish_analytics_v1.CreateSquishRatingResponse{
        SquishRatingId: uuid.New().String(),
    }
    // Save to DB

    return connect.NewResponse(newRating), nil
}
```

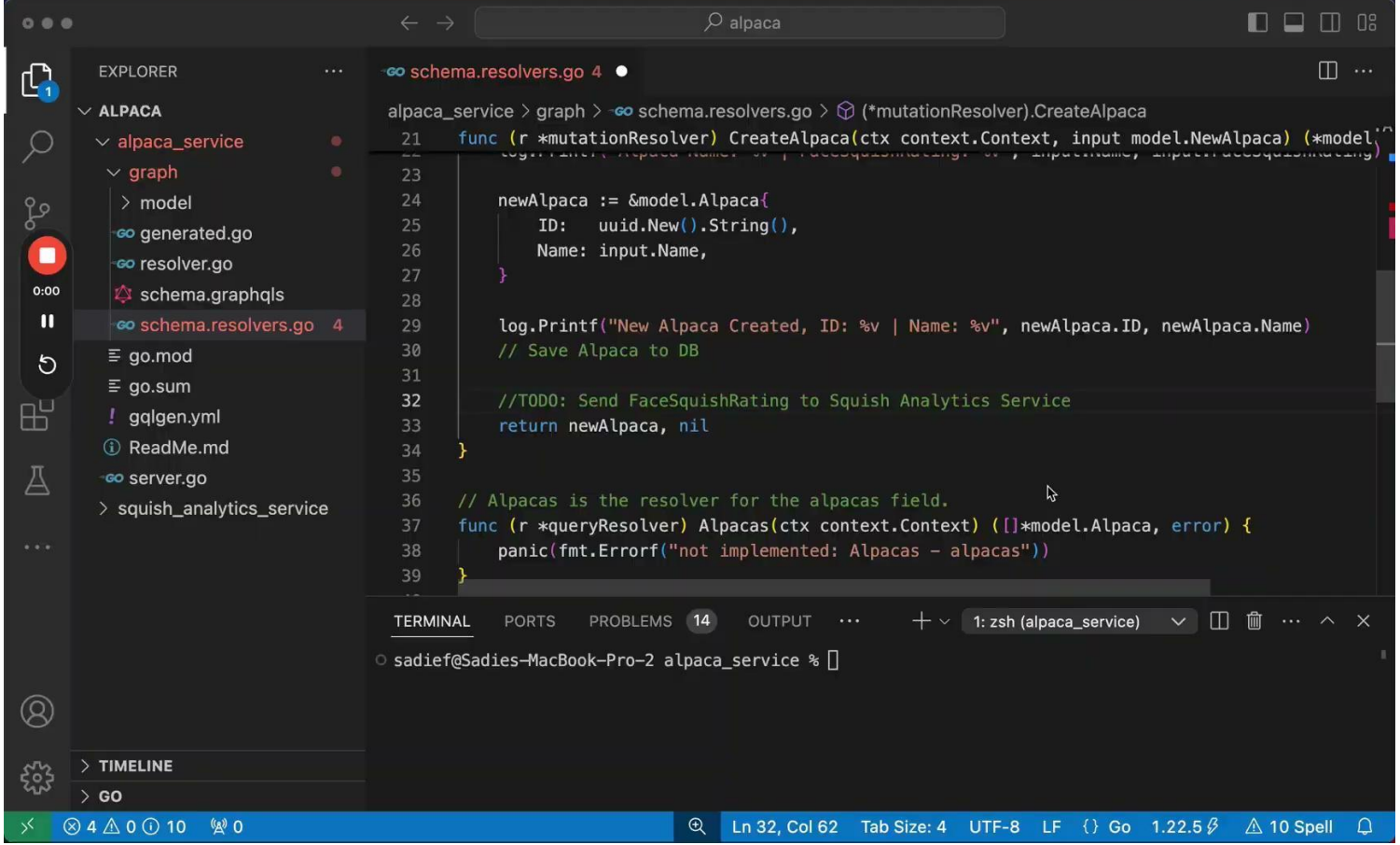

alpaca
EXPLORER
ALPACA
alpaca_service
squish-analytics-servi...
proto
squish-analytics
v1
squish-analytics.proto
server
main.go 6
0:00
main.go 6
squish-analytics.proto
squish-analytics-service > proto > squish-analytics > v1 > squish-analytics.proto
1 syntax = "proto3";
2
3 package squish_analytics.v1;
4
5 message CreateSquishRatingRequest {
6 string alpaca_id = 1;
7 string squish_rating = 2;
8 }
9
10 message CreateSquishRatingResponse {
11 string squish_rating_id = 1;
12 }
13
14 service SquishAnalyticsService {
15 rpc CreateSquishRating(CreateSquishRatingRequest) returns (CreateSquishRatingResponse) {}
16 }
17
TERMINAL
PORTS
PROBLEMS 9
OUTPUT
1: zsh (squish-analytics-se
sadief@Sadies-MacBook-Pro-2 squish-analytics-service %
TIMELINE
GO
6 0 3 0
Ln 7, Col 28
Spaces: 2
UTF-8
LF
Plain Text
1 Spell

Make them talk to each other

Buf Schema Registry (BSR)

```
buf registry login
configure buf.yaml:
 `name: buf.build/sadiefr/squishapis`)
buf push
```

```
client := squish_analyticsv1connect.NewSquishAnalyticsServiceClient(
    http.DefaultClient,
    "http://localhost:8000",
)
res, err := client.CreateSquishRating(
    context.Background(),

    connect.NewRequest(&squish_analytics_v1.CreateSquishRatingRequest{
        AlpacaId:     newAlpaca.ID,
        SquishRating: fmt.Sprintf("%v", input.FaceSquishRating),
    }),
)
```


alpaca
EXPLORER
ALPACA
alpaca_service
graph
model
generated.go
resolver.go
schema.graphqls
schema.resolvers.go 4
go.mod
go.sum
gqlgen.yml
ReadMe.md
server.go
squish_analytics_service
TIMELINE
GO
schema.resolvers.go 4
alpaca_service > graph > schema.resolvers.go > (*mutationResolver).CreateAlpaca
21 func (r *mutationResolver) CreateAlpaca(ctx context.Context, input model.NewAlpaca) (*model
24 newAlpaca := &model.Alpaca{
25 ID: uuid.New().String(),
26 Name: input.Name,
27 }
29 log.Printf("New Alpaca Created, ID: %v | Name: %v", newAlpaca.ID, newAlpaca.Name)
30 // Save Alpaca to DB
32 //TODO: Send FaceSquishRating to Squish Analytics Service
33 return newAlpaca, nil
34 }
36 // Alpacas is the resolver for the alpacas field.
37 func (r *queryResolver) Alpacas(ctx context.Context) ([]*model.Alpaca, error) {
38 panic(fmt.Errorf("not implemented: Alpacas - alpacas"))
39 }
TERMINAL
PORTS
PROBLEMS 14
OUTPUT
1: zsh (alpaca_service)
sadief@Sadies-MacBook-Pro-2 alpaca_service %
Ln 32, Col 62
Tab Size: 4
UTF-8
LF
Go
1.22.5
10 Spell

It works! Now what?

- Demo and teach
- Document the Pros and Cons
- Evaluate the time vs effort
- Present your case + alternatives

Where to go from here?

Pagination - Edges/Nodes - Caching

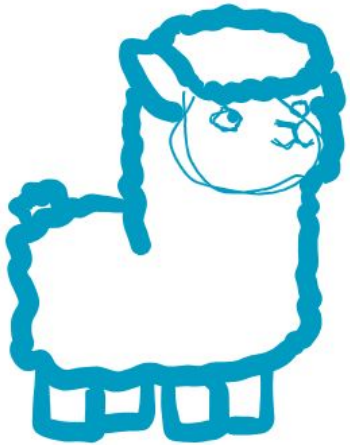
Containerization - Deployment

Event sourcing for async ops

Thank You!

Code examples + Slides

https://bit.ly/gophercon-2024



sadiefreeman@gmail.com