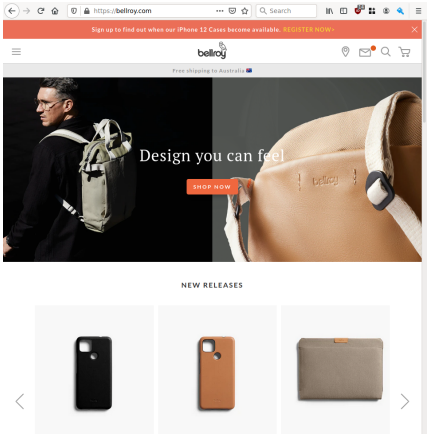
Haskell and Lambda at Bellroy

Jack Kelly

May 26, 2021

Bellroy.com



- Designs/manufactures/sells bags, wallets, backpacks, phone cases, work accessories, ...
- Internal design and technical teams maintain `bellroy.com` and do other technical work
- Over 10 years old

Where is Bellroy doing FP?

- Frontend: Elm
- Backend: FP-inspired libraries (Ruby)
- New Backend: Haskell

Where is Bellroy doing FP?

- Frontend: Elm
- Backend: FP-inspired libraries (Ruby)
- New Backend: Haskell

Haskell Backend Services

Mostly built using:

- AWS API Gateway
- AWS Lambda
- AWS DynamoDB

AWS API Gateway

- Service for defining HTTP APIs
- Integrate endpoints with various stuff
  - Including AWS Lambda Functions

AWS Lambda

"Serverless Functions-as-a-Service"

- It runs on someone else's computer
- Takes in JSON, does stuff, returns JSON

```
(FromJSON a, ToJSON b, MonadStuff m) => (a -> m b)

-- For API Gateway
MonadStuff m => ProxyRequest -> m ProxyResponse
```

AWS Lambda

"Serverless Functions-as-a-Service"

- It runs on someone else's computer
- Takes in JSON, does stuff, returns JSON
- Runtime support provided by `hal` library (among others)

```
mRuntime
  :: (FromJSON a, ToJSON b, MonadStuff m)
  => (a -> m b) -> m ()
```

Hackage has great libraries for web APIs, but we can't use them... or can we?

WAI — Web Application Interface

- Interface between web servers (e.g., `warp`) and applications
- Key type: `Application`

```
type Application
  = Request
  -> (Response -> IO ResponseReceived)
  -> IO ResponseReceived

-- Morally:
type Application' = Request -> IO Response
```

- If we can convert `ProxyRequest` to `Wai.Request` and `Wai.Response` to `ProxyResponse`, then we can lift a WAI `Application` to run on `hal`!

wai-handler-hal

```
run
  :: MonadIO m
  => Application
  -> (ProxyRequest NoAuthorizer -> m ProxyResponse)
```

Demo

Deployment

- 2 ways to deploy Lambda Functions
  - Zip
  - Docker Container Image
- Building binaries
  - `cabal` command inside a replica container: `https://hub.docker.com/r/lambci/lambda`
  - Static linking with `haskell.nix`
- Building container images
  - `nixpkgs`: `dockerTools.streamLayeredImage`

Payoffs

- Get to use the best tech on Hackage — Servant is great for APIs so let's use it.
- Payoffs from good abstractions:
  - `warp` gives better local testing
  - Can run an API Gateway off a single Lambda Function
  - Can share exeution environments for different endpoints
  - Can split a large API to manage permissions

Permission Splitting

- Lambda Functions have an "Execution Role"
- If one Lambda does everything, it'll have a lot of power
- `serve $ x :<|> y :<|> z` is an `Application`
- So are `serve x`, `serve y`, and `serve z`
- Serve subsets of your API individually, with sensible permission sets