Lambda on Lambda

Serverless Haskell on AWS

Jack Kelly

2026-05-20

What is AWS Lambda?

- “Function as a service” (FaaS)
- Run your code in the cloud
- Don’t think about servers, clusters, compute infrastructure
  - (Think about AWS complexity, IAM, and deploy pipelines instead)
- Pay per invocation, and for memory × runtime of your invocations

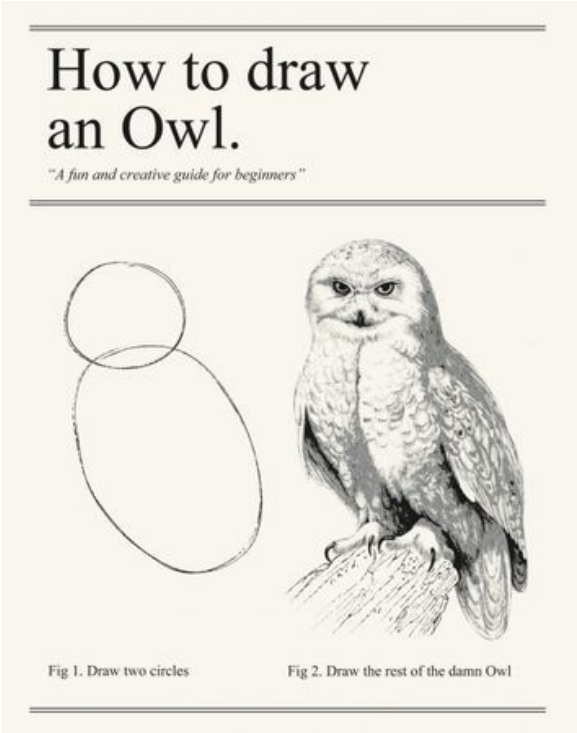
Under the marketing

- AWS uses the Firecracker MicroVM[1] to isolate your running code
- You create a "Lambda Function" from three things:
  - The code to run (`.zip` file or OCI container image)
  - The IAM "execution role" it runs under (permission to call AWS services)
  - An amount of memory to use (which linearly scales the CPU allocation)
- Invoking a lambda function sends a JSON payload via AWS API call, and your function returns JSON

[1] https://github.com/firecracker-microvm/firecracker

Tonight

- Concrete advice for running Haskell code on AWS Lambda
  - Production experience at Bellroy (~6y) mixed with my own opinions
  - Not necessarily how we currently do things
- Simple functions
- “Serverless” web services
- Working code at my lambda-on-lambda repo[1]



[1] https://git.sr.ht/~jack/lambda-on-lambda

Why use Lambda?

- Stateless
- Can be really cheap
- Many, many AWS integrations
- Event-driven workflows (trigger from S3, SQS, SNS, …)
- Elastic scaling
- Gimme a lot of compute **RIGHT NOW**
- Scale to zero
- “I just want to make a couple of API calls”

Why not use Lambda?

- Stateless
- Can be expensive
- Cold starts
- Execution environment life cycle
- Execution time limits
- Request/response payload limits
- Creeping AWS complexity

What your code must do

- Loop:
  - Ask AWS's runtime API for the next request payload
  - Process the request
  - Return the result
- `hal`[1] is a great library for this
  - Also has types for many JSON events

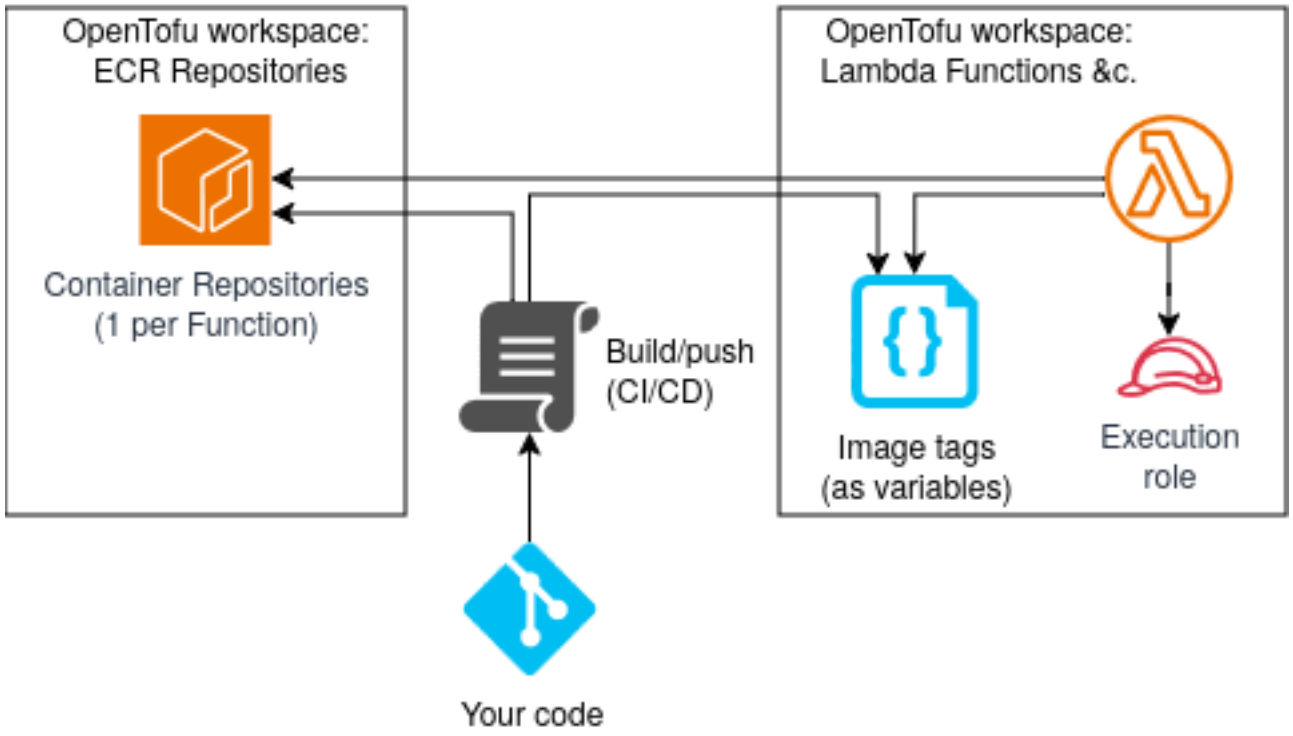
```haskell
pureRuntime
  :: (FromJSON event, ToJSON result)
  => (event -> result)
  -> IO ()

mRuntime
  :: (MonadCatch m, MonadIO m, FromJSON event, ToJSON result)
  => (event -> m result)
  -> m ()
```

[1] https://hackage.haskell.org/package/hal

Getting Haskell code into AWS

- Recommendation: Build container images, not `.zip`
- Avoids wrangling static linking and native dependencies



Testing your lambda function

- The Runtime Interface Emulator[1] implements the runtime API in a small binary
- When run by e.g. Docker, the emulator intercepts HTTP requests in lambda format and passes them to the running program
- When running on AWS, the emulator stays out of the way

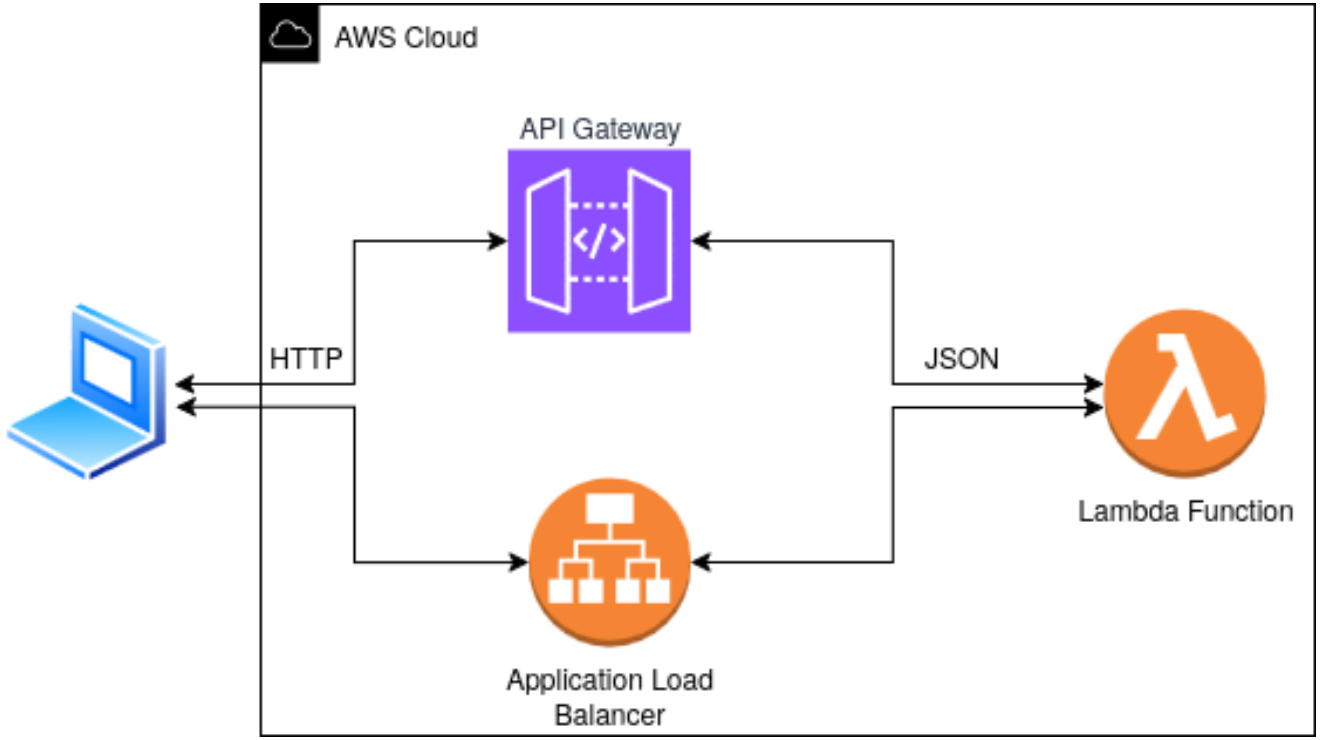
Test invocation

```shell
curl "http://localhost:9000/2015-03-31/functions/function/invocations" \
  -XPOST \
  --data '{"n": "30"}'
```

[1] https://github.com/aws/aws-lambda-runtime-interface-emulator

Serverless Web Services

- A neat way of running scalable web services
- Many AWS services integrate with Lambda by converting between HTTP (client) and JSON (lambda)



Lambda Web Adapter

- The Lambda Web Adapter[1] converts a web application to Lambda by automatically converting the JSON representation of HTTP
- Write a normal web app in whatever framework you like
  - But let's be honest it's probably gonna be `servant`
- Put the adapter in `/opt/extensions` as either a Lambda Layer (if you use `.zip`) or in your container image
  - Lambda knows to look there

[1] https://github.com/aws/aws-lambda-web-adapter

Lambda is a rabbit hole

Lambda has a lot of knobs and features that we didn't cover:

- Asynchronous events
- VPC support (access your internal network)
- EFS support (persistent storage via managed NFSv4)
- Versions/aliases
  - Pin to particular snapshots of the function
  - Split traffic between two function versions during testing
- CodeDeploy integration
  - Pre/post-traffic hooks
  - Canary deployments

Recommendations

- Check https://git.sr.ht/~jack/lambda-on-lambda
- For serious Lambda functions (not glue), Haskell is a great fit
- Use container images, not `.zip`
- For “raw” lambdas, test locally with the Runtime Interface Emulator
- For “web server” lambdas, use Haskell frameworks & Lambda Web Adapter
  - Low/scale-to-zero traffic? API Gateway HTTP API
  - High/constant traffic? Consider Application Load Balancer
- Split your infrastructure-as-code into two workspaces
  - ECR Repositories to hold the built images
  - Lambda functions to use them
- CI/CD system builds and pushes images, and reapplies the “functions” workspace with new variables