1. **Learn 'Writing an AWS lambda function with Golang'**

http://www.avitzurel.com/blog/2016/06/17/writing-an-aws-lambda-function-with-golang/

AWS lambda supports Python, Node.js, Java and C# directly.
AWS Lambda functions can support Golang with a thin wrapper on Node.js.
Example:
Lambda function that will be a subscriber to SNS notification.
AWS SNS (Simple Notification Service) Message -> NodeJS Wrapper (Node code) -> Go code

Next step: run this example

2. **Learn 'A Go framework for AWS Lambda - Sparta'**

https://medium.com/@mweagle/a-go-framework-for-aws-lambda-ab14f0c42cb
AWS Lambda's advantage: FaaS (Function as a service)

Sparta – an application written in Go (http://gosparta.io/)

Go's advantage - offers a solid toolchain and set of primitives to write services:

- Single binary deployment
- Excellent concurrency primitives

https://blog.golang.org/pipelines
Go's concurrency primitives make it easy to construct streaming data pipelines that make efficient use of I/O and multiple CPUs.

- An official AWS Go SDK

https://aws.amazon.com/sdk-for-go/?nc1=h_ls
Install the SDK for all services:
go get github.com/aws/aws-sdk-go/...
import "github.com/aws/aws-sdk-go/service/lambda"

Package lambda provides a client for AWS Lambda.

- Extremely fast compilation
- Well-defined error handling patterns

https://blog.golang.org/error-handling-and-go
Go code uses error values to indicate an abnormal state. For example, the os.Open function returns a non-nil error value when it fails to open a file.

- Static types
- Minimal startup overhead
- Cross-platform compilation
- Rich standard library

https://golang.org/pkg/

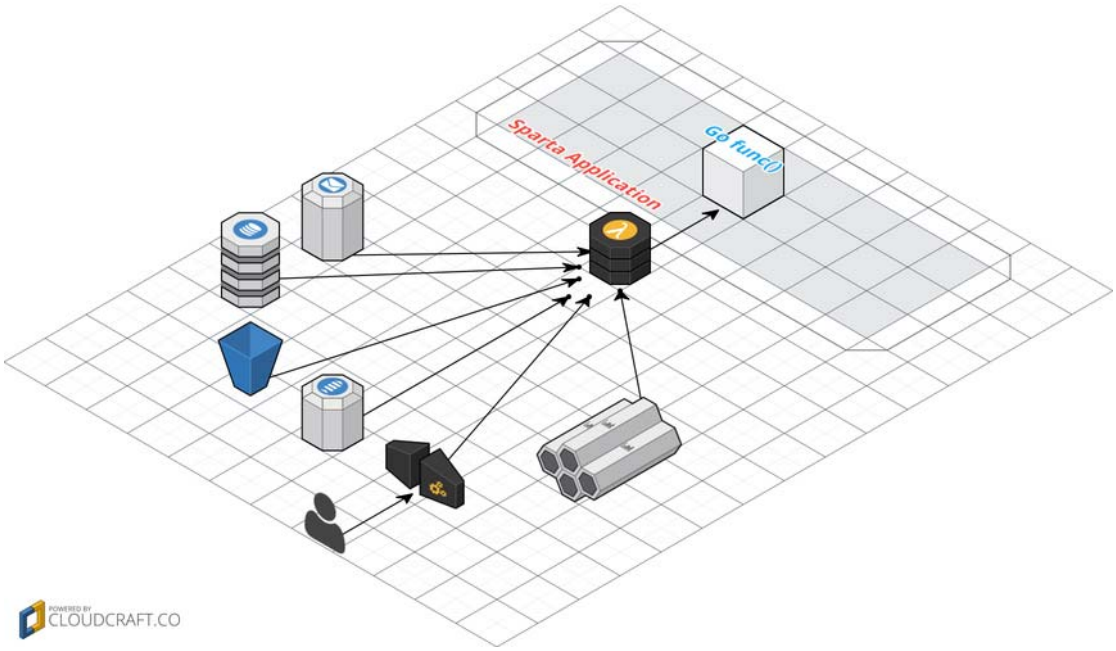
- It's boring, in the best possible way

http://stevebate.silvrback.com/go-is-boring
plain, safe, sometimes backwards

Given a little help from one of the officially supported runtimes, a compiled Go binary could be the target of an AWS Lambda request.

Example: HTTP All The Things
For each Sparta-compatible AWS Lambda function (defined below), Sparta creates a unique NodeJS proxy route to forward the AWS request to a sidecar Go binary.

- Cross-compiles the Go binary for AWS's Linux
- Dynamically creates a NodeJS HTTP proxy entry for each unique Sparta lambda function. Each entry represents an addressable AWS Lambda function.
- Builds a deployable ZIP archive that includes the Go binary and dynamically created JS contents.
- Creates a CloudFormation template for provisioning, using content-based resource names
- Either creates or updates CloudFormation stack



Next step: run the Sparta example

3. FaaS concept

https://github.com/alexellis/faas/
http://blog.alexellis.io/functions-as-a-service/
http://docs.aws.amazon.com/lambda/latest/dg/welcome.html
FaaS is a framework for building serverless functions on Docker Swarm with first class support for metrics. Any UNIX process can be packaged as a function enabling you to consume a range of web events without repetitive boiler-plate

coding.

- tends involve invoking short-lived functions (Lambda has a default 1-sec timeout)
- does not publish TCP services - often accesses third-party services or resources
- tends to be ad-hoc/event-driven such as responding to webhooks
- should deal gracefully with spikes in traffic
- despite name, runs on servers backed by a fluid, non-deterministic infrastructure
- when working well makes the complex appear simple
- has several related concerns: infrastructure management, batching, access control, definitions, scaling & dependencies

We can build serverless applications composed of functions that are triggered by events and automatically deploy them using AWS CodePipeline and AWS CodeBuild.

4. Use AWS Lambda

5. Example code of running Go code on AWS Lambda