Stop writing code that only runs on AWS Lambda

Stop drinking the koolaid.

AWS Lambda?

How many people have heard/used AWS Lambda?

A fully managed, Platform as a Service. You provide code, Lambda will run it.

Like any Platform as a Service, there are pros and cons.

Pros

- Autoscale up and down
- Critical and Security OS Updates handled automatically
- Centralized Logging
- Centralized Monitoring
- No servers

All of these things are built in.

It’s an easy way to have production grade infrastructure without the standard costs or resource allocation.

Cons

- AWS Lambda can require code changes.

- As soon as we start to write custom code for the platform we're leveraging we're locking ourselves into that platform.
- The point of this talk is to show how we can mitigate the impact of having to write custom code for AWS Lambda.
- We still have to write the custom code but we can do it in an isolated way to not lock ourselves into AWS Lambda
- Let's look at some code.

Standard Go

- We’ll start with standard go

Standard Go

```
package main

import (
  "fmt"
)

func main() {
  fmt.Println("Hello World!")
}
```

- Here's how you'd start a typical go http server.
- *Read file from top*
- This should be able to run anywhere go runs.

AWS Lambda Go

AWS Lambda Go

```go
package main

import (
  "context"
  "fmt"
  "github.com/aws/aws-lambda-go/lambda"
)

func HandleRequest(ctx context.Context) error {
  fmt.Println("Hello World")
  return nil
}

func main() {
  lambda.Start(HandleRequest)
}
```

- Here’s how you’d start a lambda http server.
- *Read file from top.*

Putting them together

So now we've seen the slightly different code needed for standalone and aws lambda.

What if we can put them all together, so we can leverage any of these environments with one code base.

Putting them all together

- main.go
- main_aws_lambda.go
- main_standalone.go

- We’re going to create 3 files.
- *Read names of files.*
- Each of these files will implement the environment specific code then call a universal main func that will execute all the non environment specific code.
- Let’s look at the code for each of these files.

main.go

```
package main

import (
  “fmt"
)

func DoSomething() {
  fmt.Println("Hello World!")
}
```

Where we define our universal Main func. This function will be called by each environments main func.

main_aws_lambda.go

```go
// +build aws_lambda

package main

import (
  "context"
  "fmt"
  "github.com/aws/aws-lambda-go/lambda"
)

func HandleRequest(ctx context.Context) error {
  DoSomething()
  return nil
}

func main() {
  lambda.Start(HandleRequest)
}
```

- Build tags list the conditions under which a file should be included in the package during the build.
- Read file from top

main_standalone.go

```
// +build !aws_lambda

package main

func main() {
 DoSomething()
}
```

- *Read file from top*

Questions