SERVERLESS EM GO

Elton Minetto

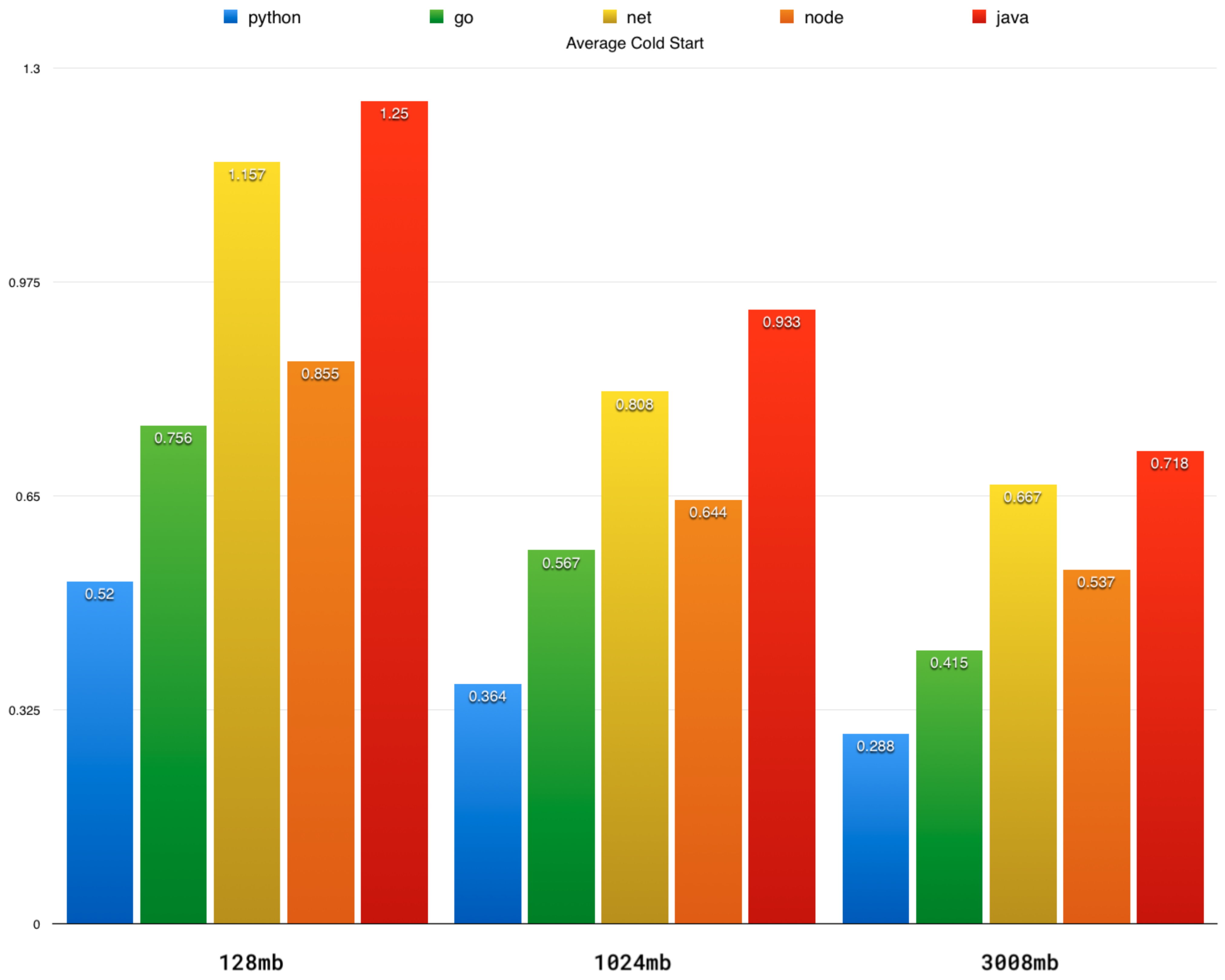
@eminetto

Five Key Benefits of "Going Serverless"

1. Auto Scales for you 📈
2. Pay per execution pricing model 💸
3. Leverage third party services 🤝
4. Focus on your business logic 😁
5. Event driven (push based) workflows 🚀

https://www.netlify.com/blog/2018/08/06/five-key-benefits-of-going-serverless/#what-are-the-benefits-of-going-serverless

Por que usar Go em suas Lambda functions?



https://medium.com/@nathan.malishev/lambda-cold-starts-language-comparison-%EF%B8%8F-a4f4b5f16a62

Nossas experiências

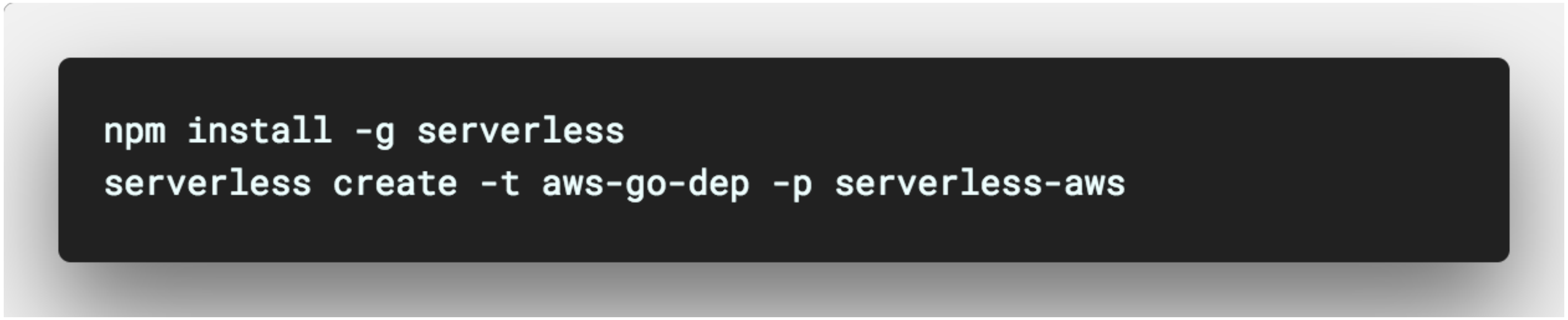
Por que usar frameworks?

- Independência de fornecedor de nuvem
- Facilidade de deploy automatizado
- Integração/padronização com outras linguagens do projeto

Serverless Framework

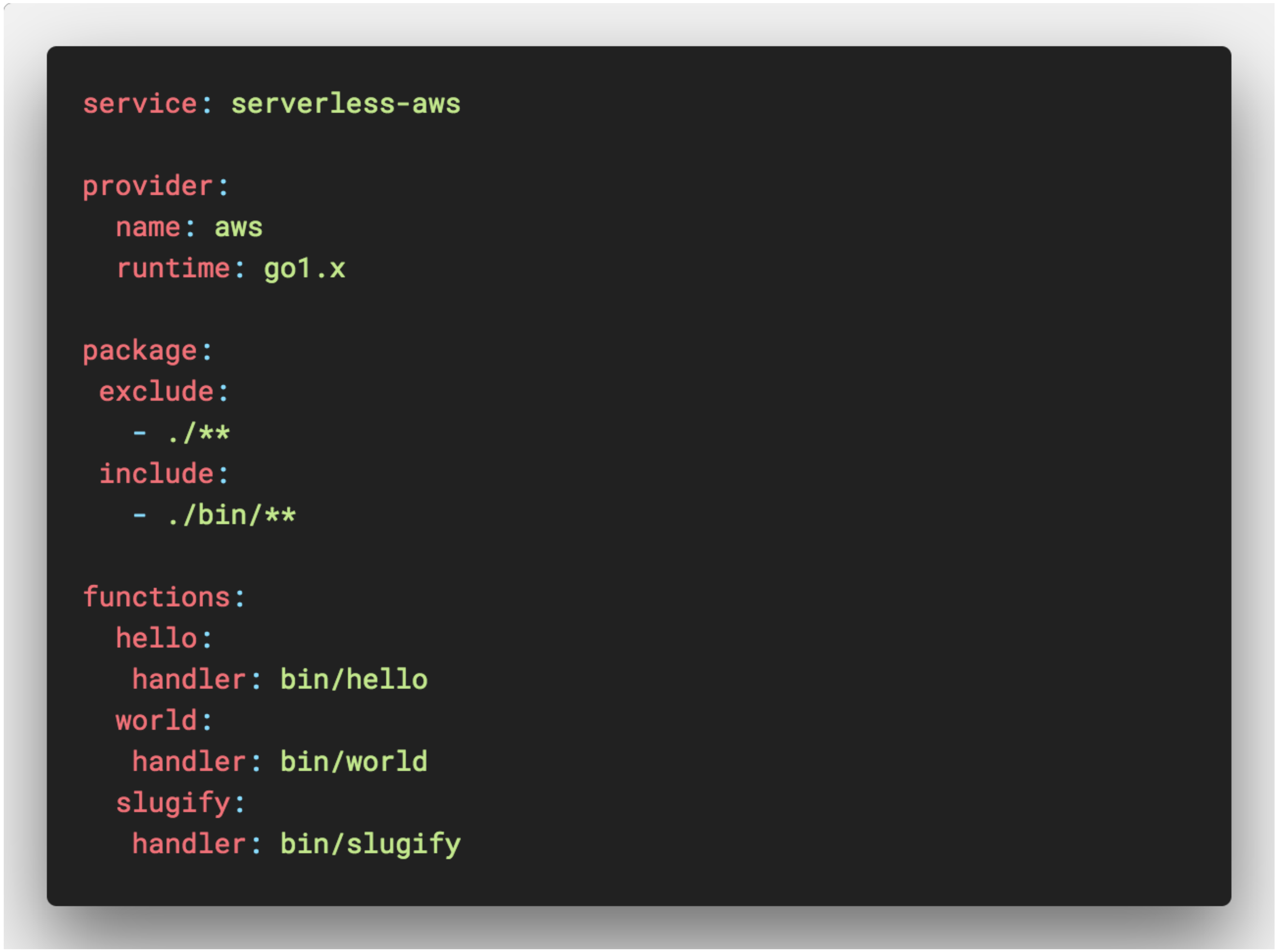
- Suporte a várias linguagens além do Go
- Suporte a múltiplos providers (em Go por enquanto apenas AWS e fn)
- Suporte a eventos
- Diversos plugins
- Requer uso dos pacotes dos providers (AWS, Google, etc)
- A empresa recentemente recebeu um aporte de capital

AWS



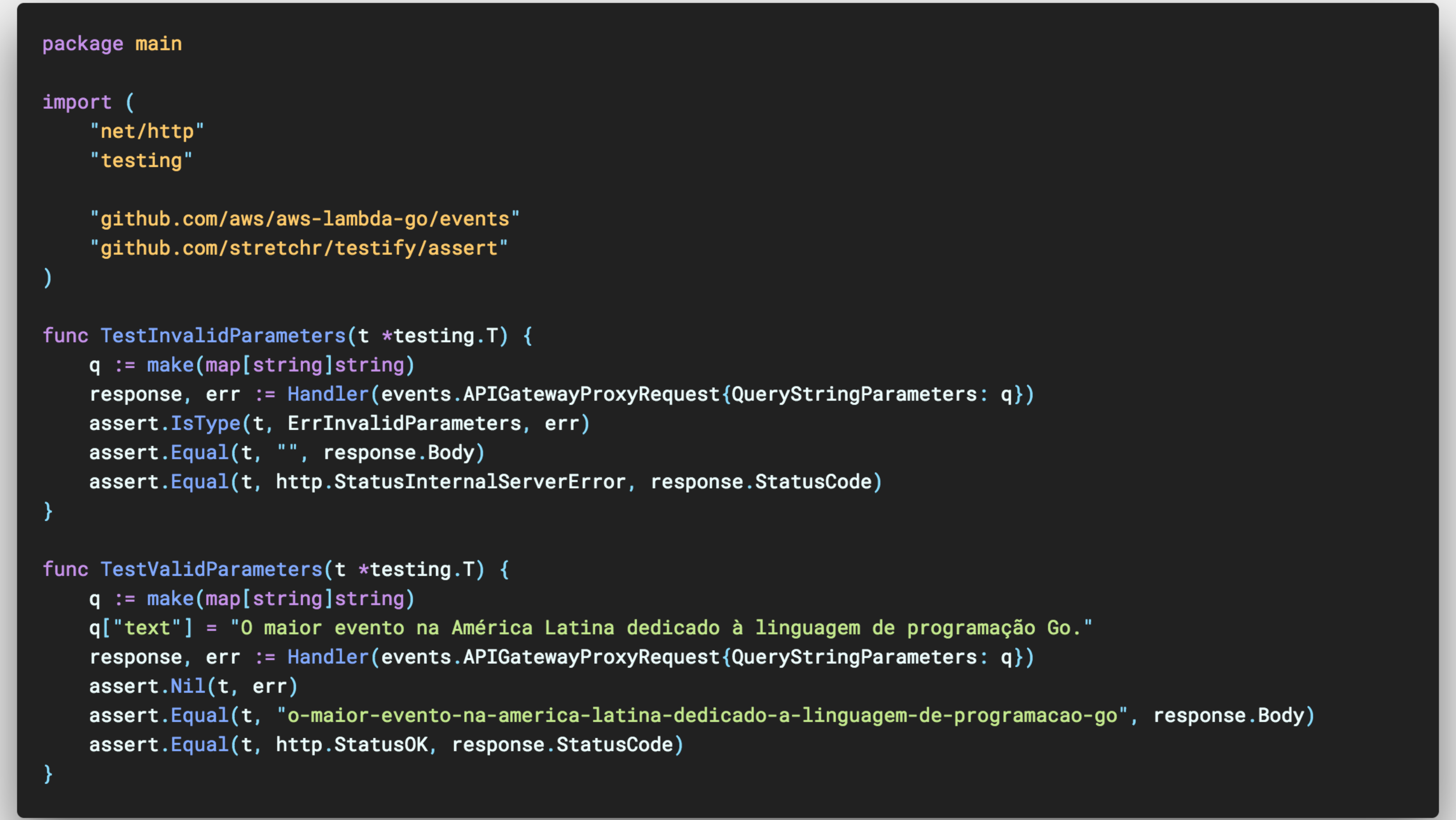
```
npm install -g serverless
serverless create -t aws-go-dep -p serverless-aws
```

Configurar o arquivo `serverless.yml`



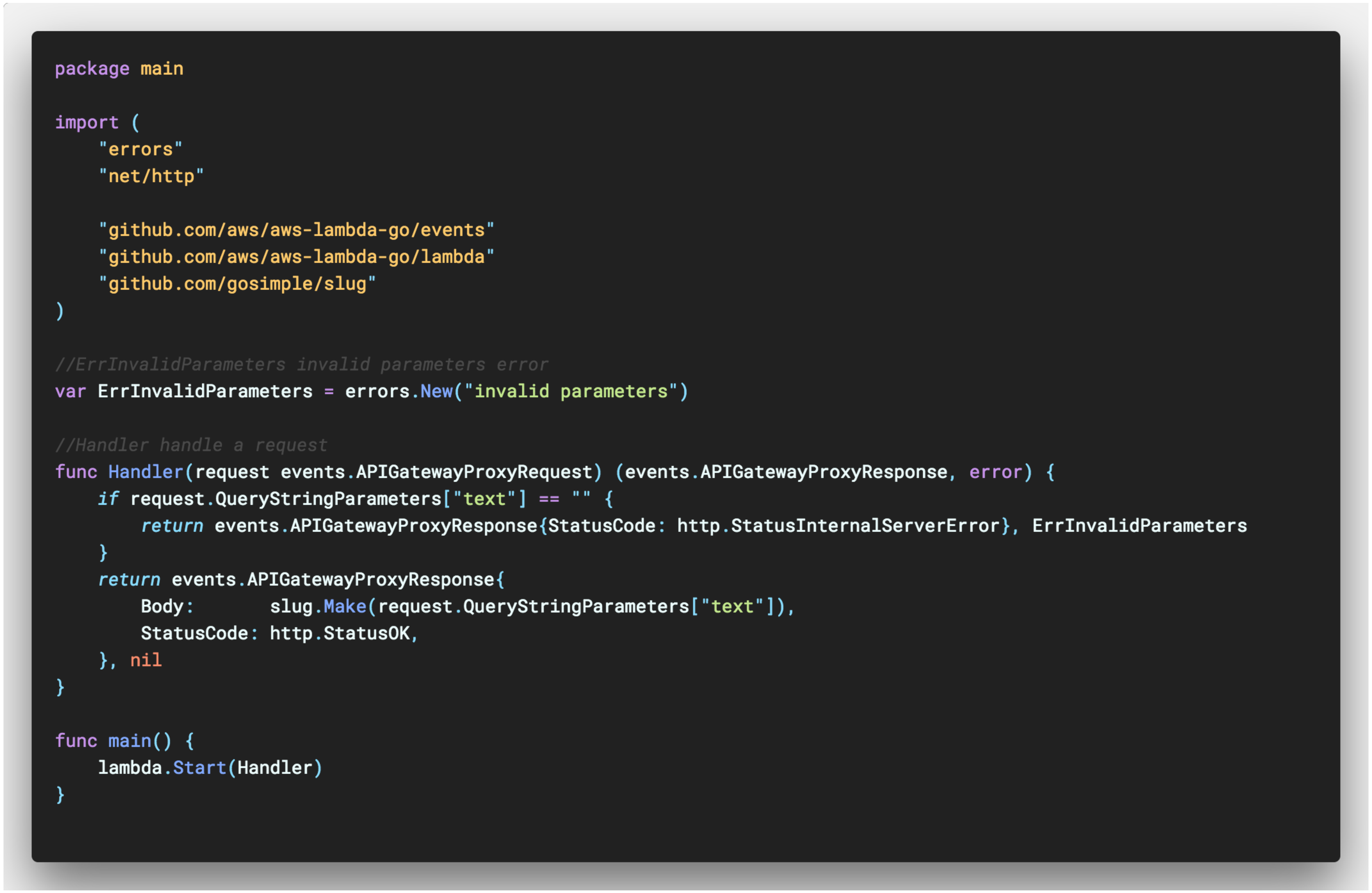
```yaml
service: serverless-aws

provider:
  name: aws
  runtime: go1.x

package:
 exclude:
   - ./**
 include:
   - ./bin/**

functions:
  hello:
   handler: bin/hello
  world:
   handler: bin/world
  slugify:
   handler: bin/slugify
```

slugify/main_test.go



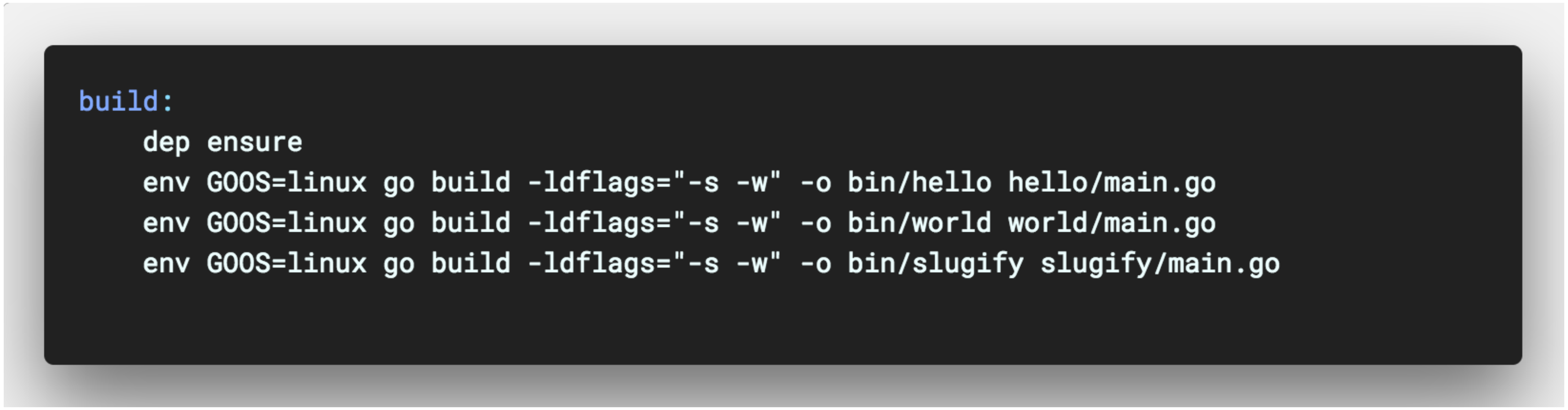
```go
package main

import (
    "net/http"
    "testing"

    "github.com/aws/aws-lambda-go/events"
    "github.com/stretchr/testify/assert"
)

func TestInvalidParameters(t *testing.T) {
    q := make(map[string]string)
    response, err := Handler(events.APIGatewayProxyRequest{QueryStringParameters: q})
    assert.IsType(t, ErrInvalidParameters, err)
    assert.Equal(t, "", response.Body)
    assert.Equal(t, http.StatusInternalServerError, response.StatusCode)
}

func TestValidParameters(t *testing.T) {
    q := make(map[string]string)
    q["text"] = "O maior evento na América Latina dedicado à linguagem de programação Go."
    response, err := Handler(events.APIGatewayProxyRequest{QueryStringParameters: q})
    assert.Nil(t, err)
    assert.Equal(t, "o-maior-evento-na-america-latina-dedicado-a-linguagem-de-programacao-go", response.Body)
    assert.Equal(t, http.StatusOK, response.StatusCode)
}
```

slugify/main.go



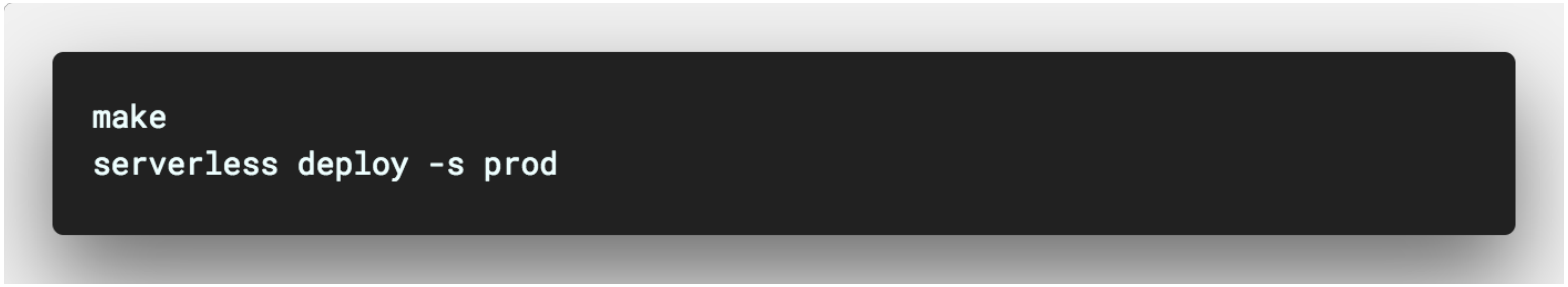
```go
package main

import (
    "errors"
    "net/http"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/gosimple/slug"
)

//ErrInvalidParameters invalid parameters error
var ErrInvalidParameters = errors.New("invalid parameters")

//Handler handle a request
func Handler(request events.APIGatewayProxyRequest) (events.APIGatewayProxyResponse, error) {
    if request.QueryStringParameters["text"] == "" {
        return events.APIGatewayProxyResponse{StatusCode: http.StatusInternalServerError}, ErrInvalidParameters
    }
    return events.APIGatewayProxyResponse{
        Body:       slug.Make(request.QueryStringParameters["text"]),
        StatusCode: http.StatusOK,
    }, nil
}

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

Makefile



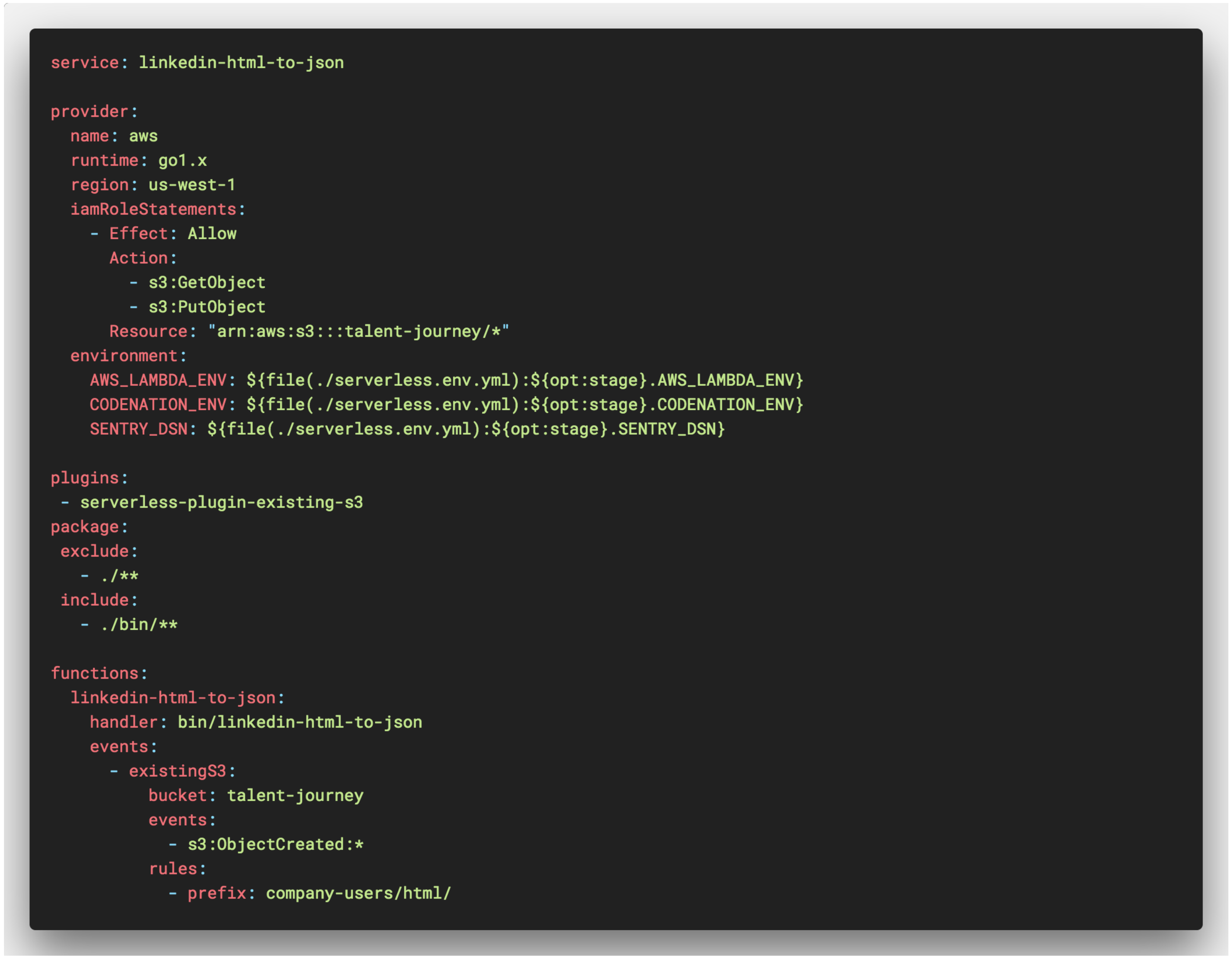
```
build:
	dep ensure
	env GOOS=linux go build -ldflags="-s -w" -o bin/hello hello/main.go
	env GOOS=linux go build -ldflags="-s -w" -o bin/world world/main.go
	env GOOS=linux go build -ldflags="-s -w" -o bin/slugify slugify/main.go
```

Deploy



```
make
serverless deploy -s prod
```

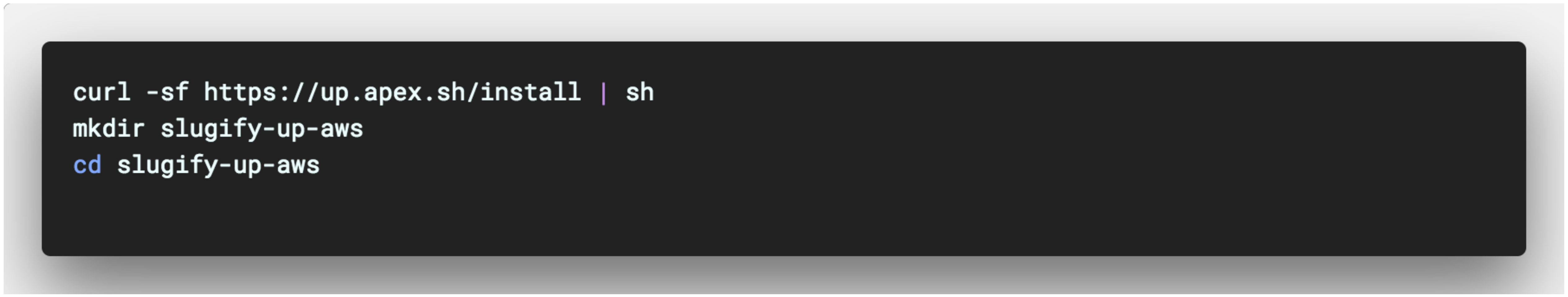
Exemplo usando outros eventos, plugins e variáveis de ambiente



```yaml
service: linkedin-html-to-json

provider:
  name: aws
  runtime: go1.x
  region: us-west-1
  iamRoleStatements:
    - Effect: Allow
      Action:
        - s3:GetObject
        - s3:PutObject
      Resource: "arn:aws:s3:::talent-journey/*"
  environment:
    AWS_LAMBDA_ENV: ${file(./serverless.env.yml):${opt:stage}.AWS_LAMBDA_ENV}
    CODENATION_ENV: ${file(./serverless.env.yml):${opt:stage}.CODENATION_ENV}
    SENTRY_DSN: ${file(./serverless.env.yml):${opt:stage}.SENTRY_DSN}

plugins:
 - serverless-plugin-existing-s3
package:
 exclude:
   - ./**
 include:
   - ./bin/**

functions:
  linkedin-html-to-json:
    handler: bin/linkedin-html-to-json
    events:
      - existingS3:
          bucket: talent-journey
          events:
            - s3:ObjectCreated:*
          rules:
            - prefix: company-users/html/
```

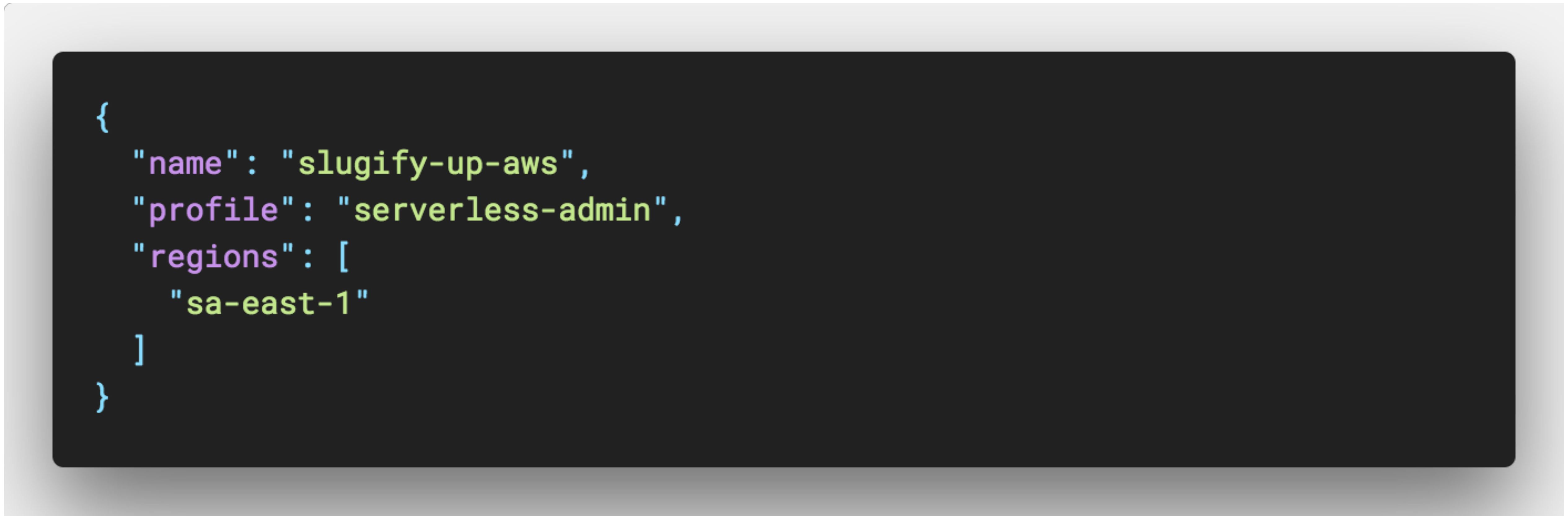
up

- Suporte a várias linguagens além do Go
- Atualmente suporte apenas a AWS, Google e Azure em desenvolvimento
- Desenvolvimento e deploy rápido para criar lambdas acessadas via HTTP
- Usa apenas a sdtlib



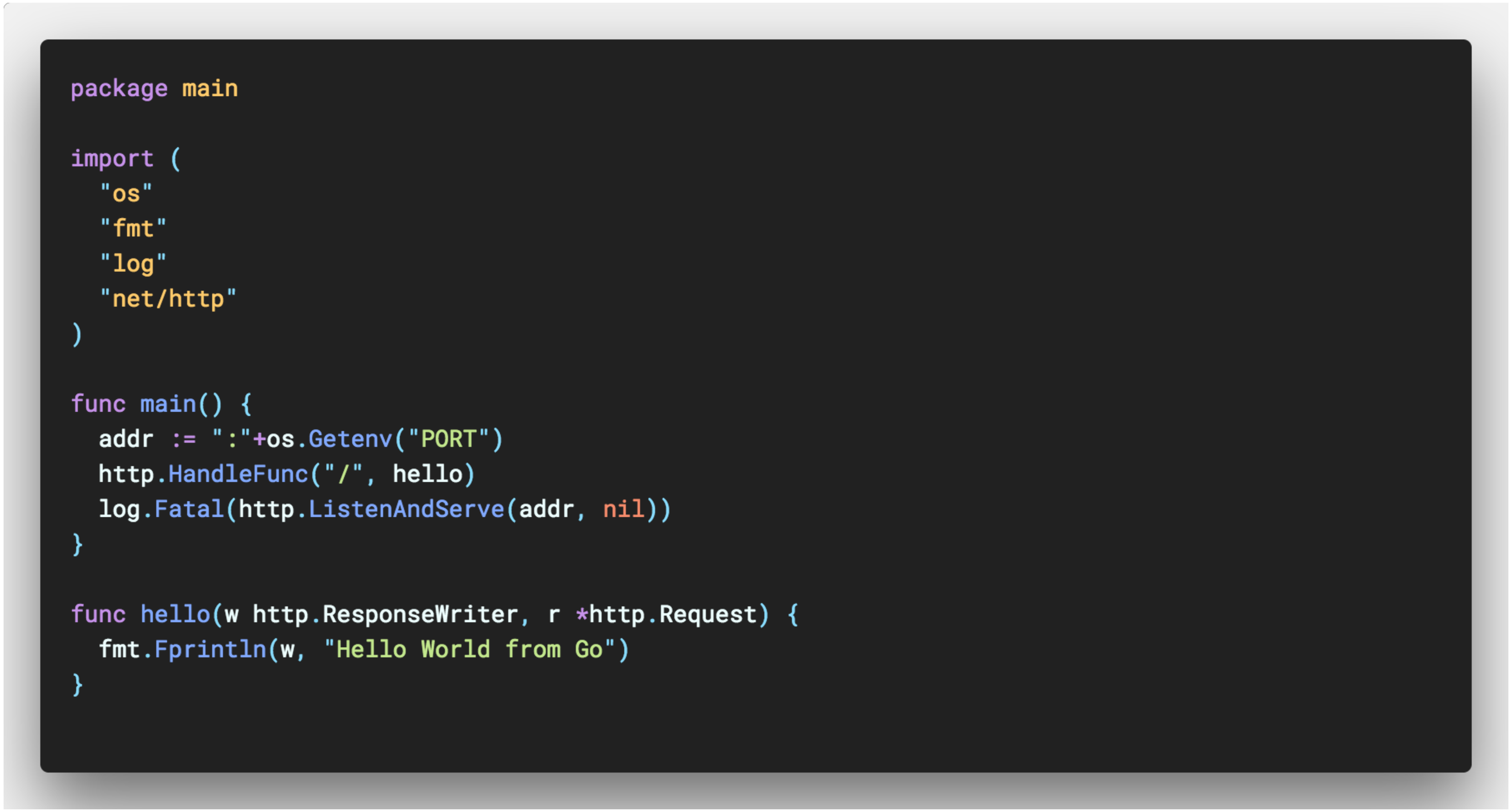
```
curl -sf https://up.apex.sh/install | sh
mkdir slugify-up-aws
cd slugify-up-aws
```

Configurações são feitas no `up.json`



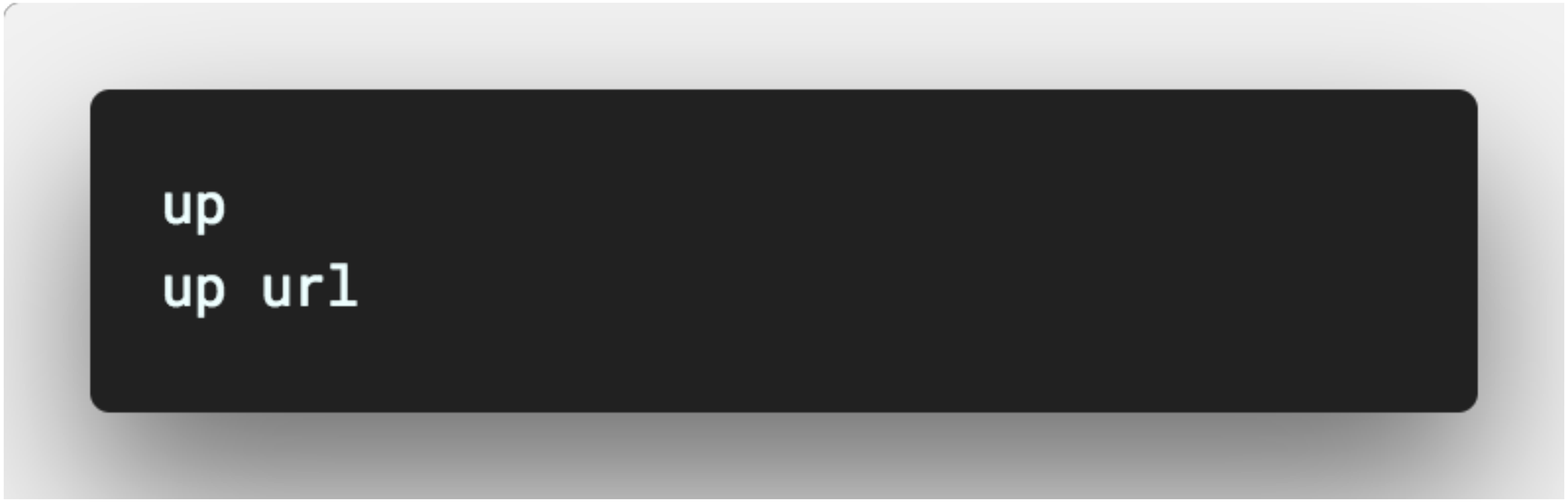
```json
{
  "name": "slugify-up-aws",
  "profile": "serverless-admin",
  "regions": [
    "sa-east-1"
  ]
}
```

Criar um arquivo `main.go`:



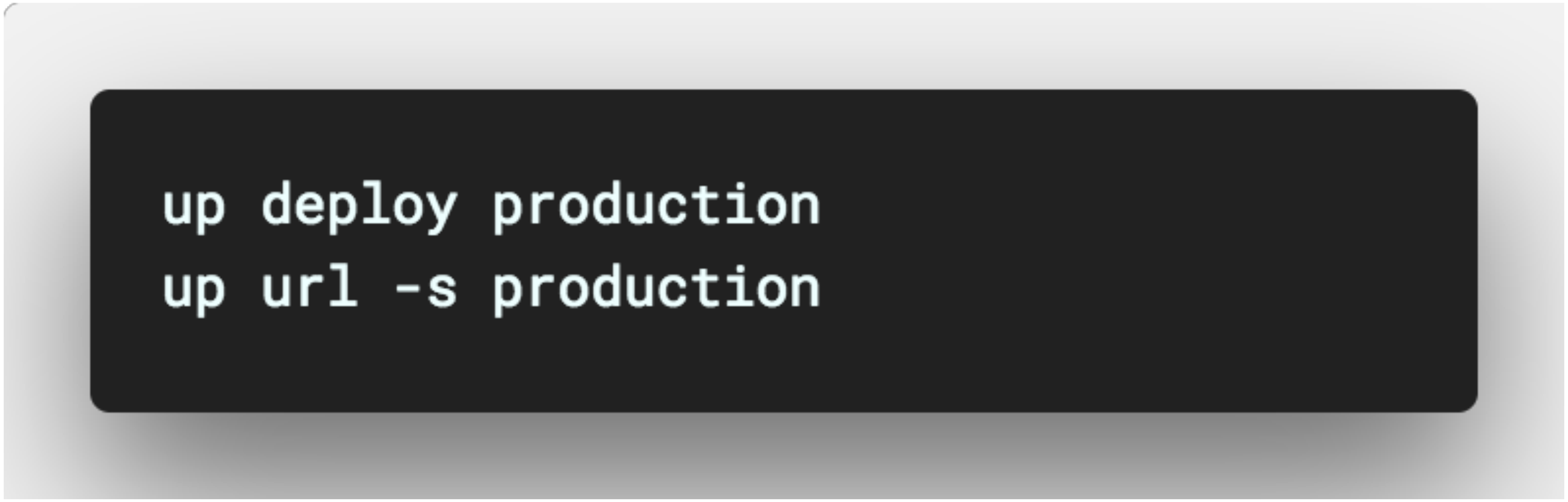
```go
package main

import (
  "os"
  "fmt"
  "log"
  "net/http"
)

func main() {
  addr := ":"+os.Getenv("PORT")
  http.HandleFunc("/", hello)
  log.Fatal(http.ListenAndServe(addr, nil))
}

func hello(w http.ResponseWriter, r *http.Request) {
  fmt.Fprintln(w, "Hello World from Go")
}
```

Compilar e enviar para o ambiente de staging



```
up
up url
```

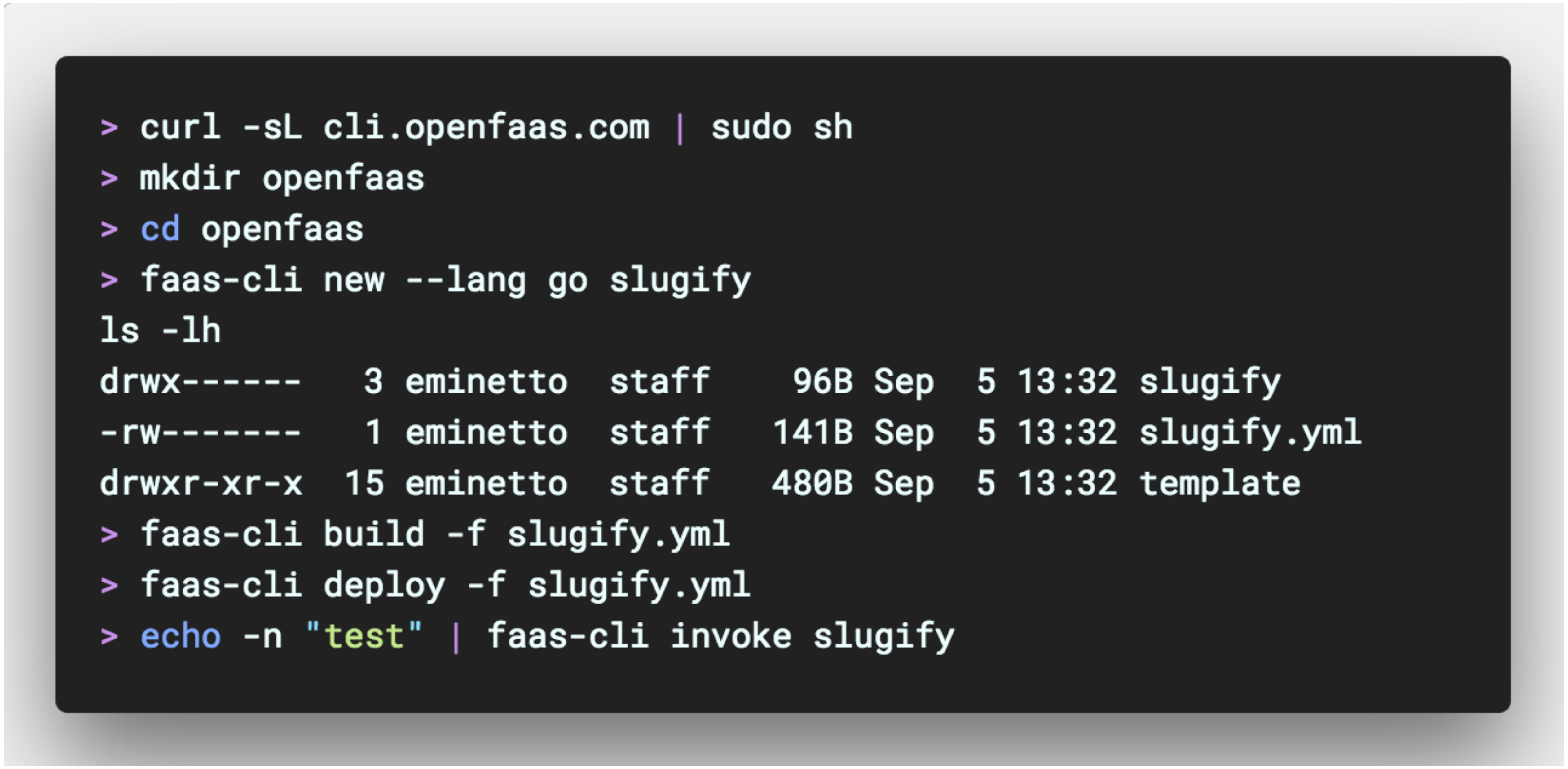
Fazer o deploy para produção



```
up deploy production
up url -s production
```

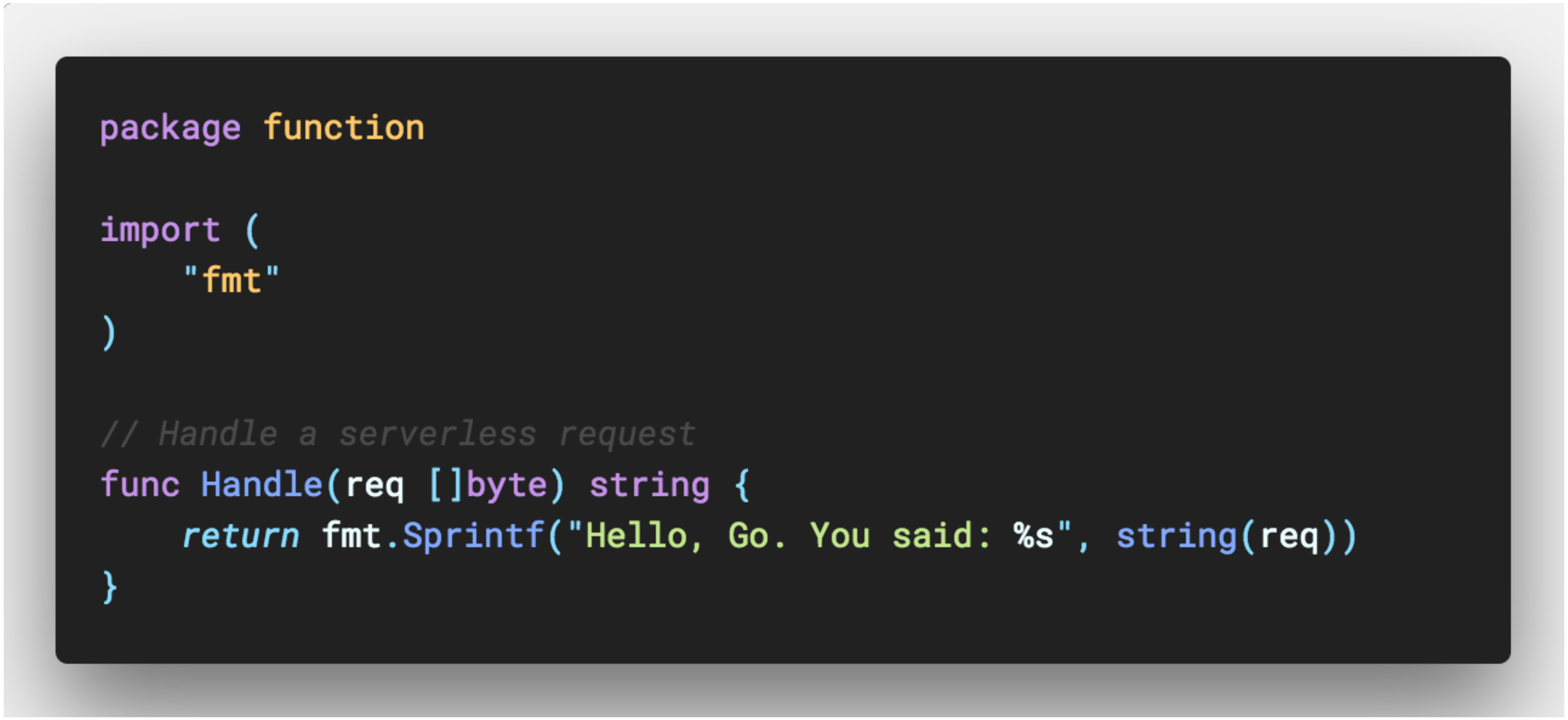
OpenFaas

- Suporte a várias linguagens além do Go
- Independente de provider
- Baseado em containers Docker/ Kubernetes



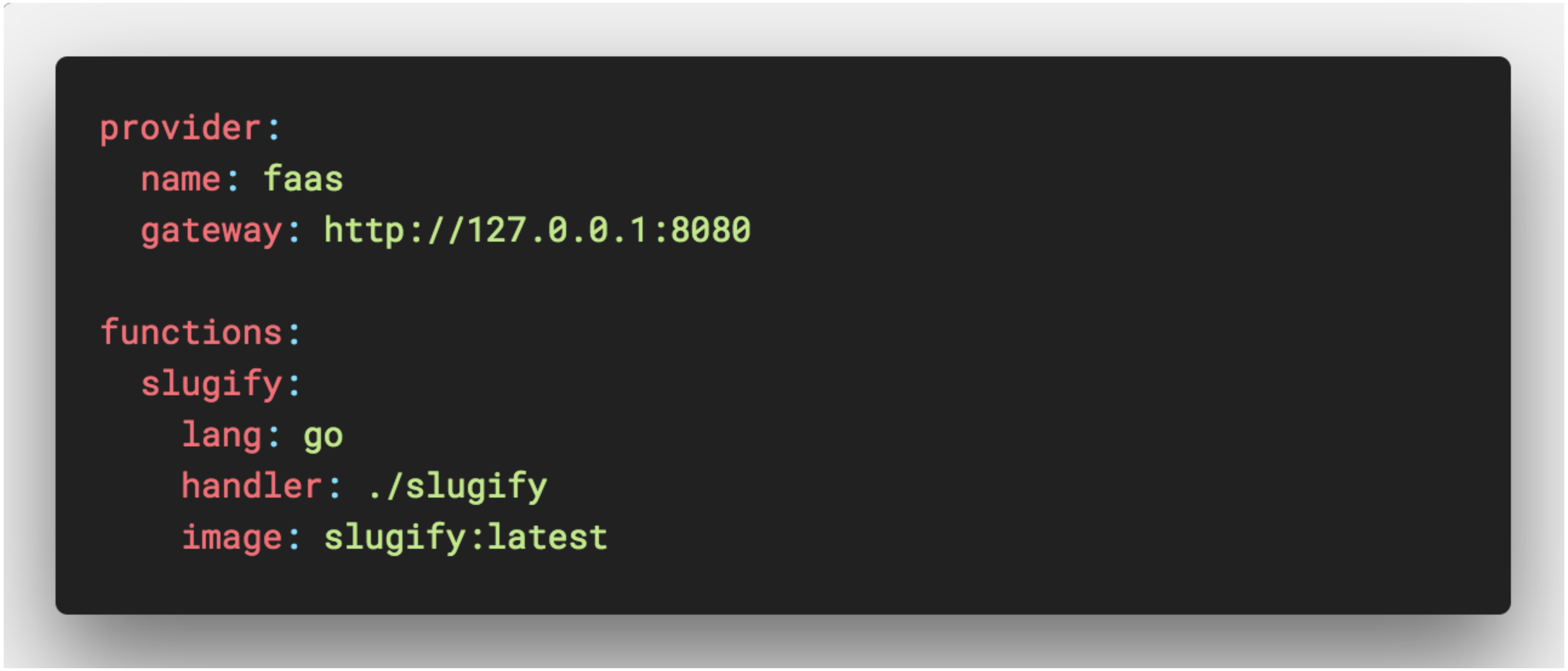
```
> curl -sL cli.openfaas.com | sudo sh
> mkdir openfaas
> cd openfaas
> faas-cli new --lang go slugify
ls -lh
drwx------   3 eminetto  staff    96B Sep  5 13:32 slugify
-rw-------   1 eminetto  staff   141B Sep  5 13:32 slugify.yml
drwxr-xr-x  15 eminetto  staff   480B Sep  5 13:32 template
> faas-cli build -f slugify.yml
> faas-cli deploy -f slugify.yml
> echo -n "test" | faas-cli invoke slugify
```

slugify/handler.go



```go
package function

import (
	"fmt"
)

// Handle a serverless request
func Handle(req []byte) string {
	return fmt.Sprintf("Hello, Go. You said: %s", string(req))
}
```

slugify.yml



```yaml
provider:
  name: faas
  gateway: http://127.0.0.1:8080

functions:
  slugify:
    lang: go
    handler: ./slugify
    image: slugify:latest
```

Menção honrosa

https://github.com/gofn/gofn

Qual escolher?

- Se precisa tratar tipos diferentes de eventos e conviver com outras linguagens: Serverless
- Se precisa apenas criar lambdas para APIs: Up
- Se quer ficar independente de fornecedor e usar sua infra de containers: OpenFaas/gofn

Links

https://gist.github.com/eminetto/a4dda9d66ff42aece666af0903672d6b

Perguntas

http://eltonminetto.net

https://www.codenation.com.br

@eminetto

http://asemanago.com.br