Serverless is not magic

or is it?

TL;DR it isn't

Duarte Mendes
Paulo Andrade

26/11/19 @Mindera

Disclaimer

This is merely our own experience

Typical application

Exposes one port

That port has a HTTP server listening to requests

This server exposes several routes

We need a container orchestrator

Manage infrastructure (scaling and concurrency)

Lambdas

Service that provision and manages the servers (scalability and concurrency)

Export a function with the code

Pay only for what you use (time, memory and request)

Advantages

No management of server hosts or server processes

Costs based on precise usage

Implicit high availability

Self auto-scale and auto-provision based on load

Performance capabilities defined in terms other than host size/count

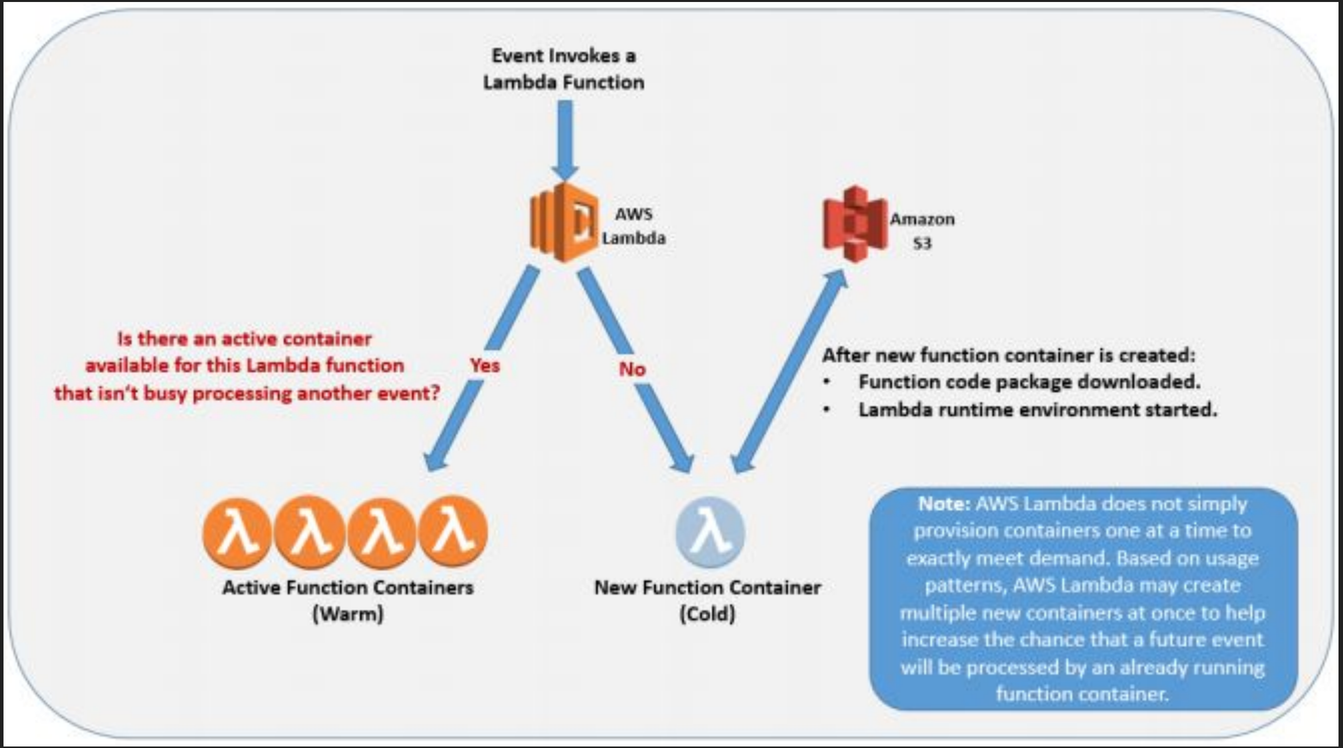
Features branches out of the box

Faster development

What if we told you...

Lambdas run on containers

Concurrency


Event Invokes a Lambda Function
AWS Lambda
Amazon S3
Is there an active container available for this Lambda function that isn't busy processing another event?
Yes
No
Active Function Containers (Warm)
New Function Container (Cold)

After new function container is created:

- Function code package downloaded.
- Lambda runtime environment started.

Note: AWS Lambda does not simply provision containers one at a time to exactly meet demand. Based on usage patterns, AWS Lambda may create multiple new containers at once to help increase the chance that a future event will be processed by an already running function container.

Concurrency & Scaling

Default concurrency limit: 1000

Initial burst: up to 500 - 3000 depending on region

After that: scales up to 500 instances a minute

When the number of requests decreases, unused instances are stopped

Concurrency limits are shared between all lambdas in an AWS account and region

Lambda Execution Context

Provisioned by aws based on the function configuration

Temporary runtime environment

It’s reused in the next invocations

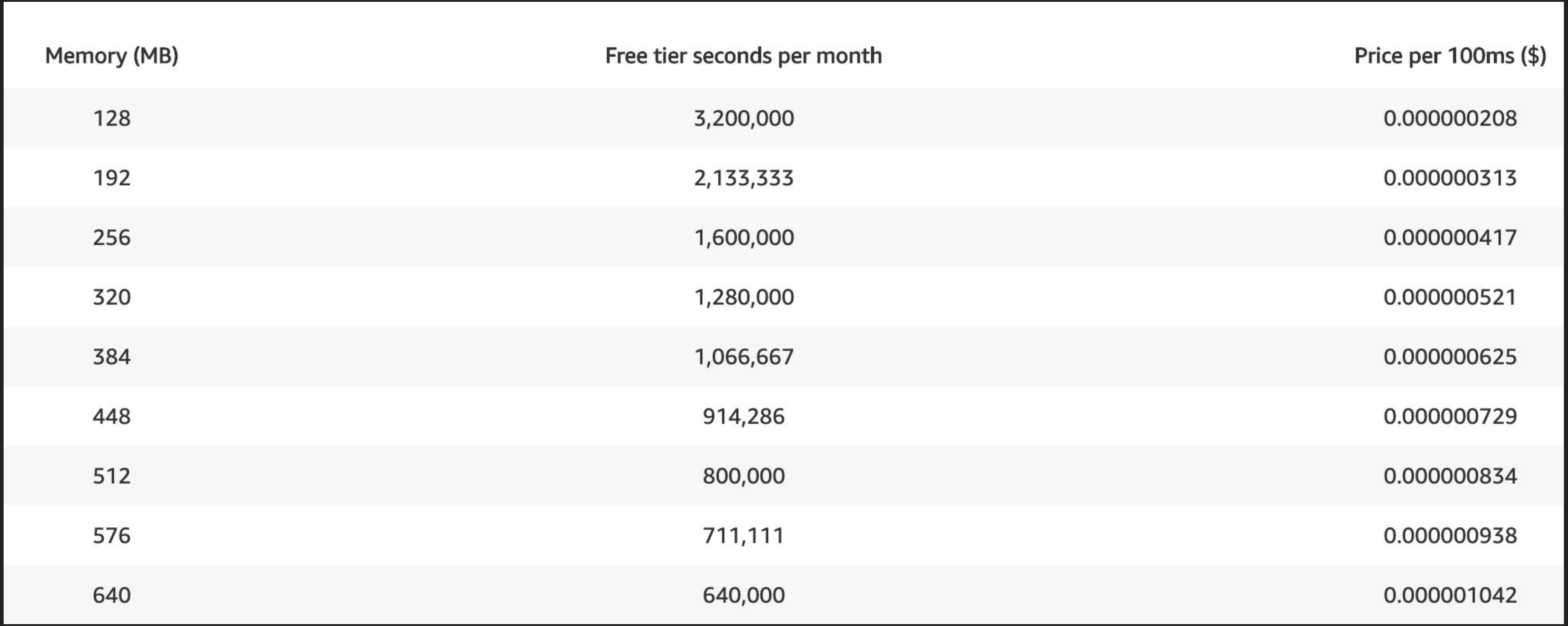
Similar to a docker startup of a container

Available runtimes

- Node.js - 12, 10, 8.10
- Python - 3.8, 3.7, 3.6, 2.7
- Ruby - 2.5
- Java - 11, 8
- Go - 1.x
- .NET Core - 2.1

(24/11/19)

Pricing



Memory (MB)	Free tier seconds per month	Price per 100ms ($)
128	3,200,000	0.000000208
192	2,133,333	0.000000313
256	1,600,000	0.000000417
320	1,280,000	0.000000521
384	1,066,667	0.000000625
448	914,286	0.000000729
512	800,000	0.000000834
576	711,111	0.000000938
640	640,000	0.000001042

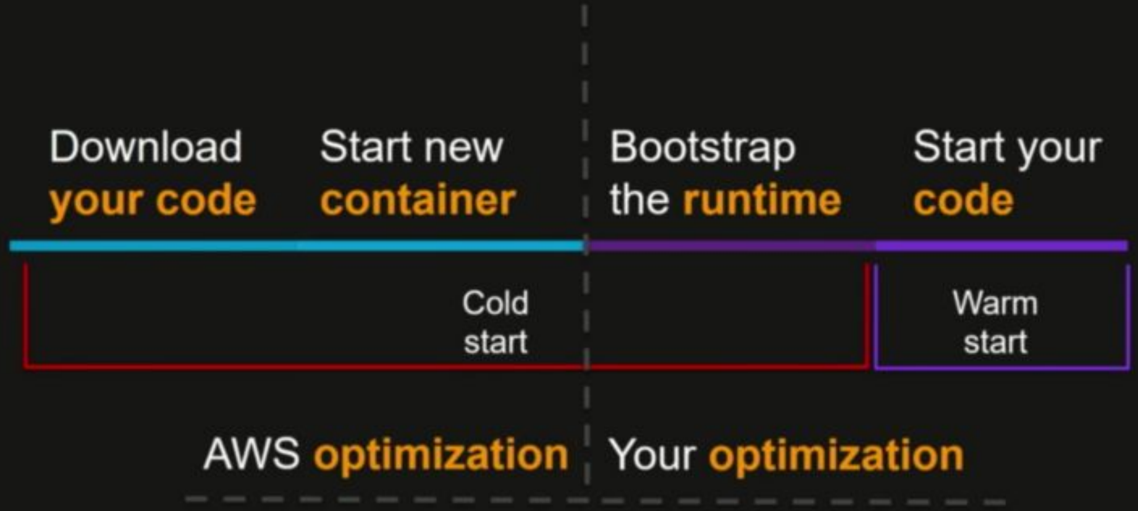
Automatic retries and DLQ

Up to 2 automatic retries on failed asynchronous invocations

When all retries fail, send the event to a DQL:

- SQS queue
- SNS topic

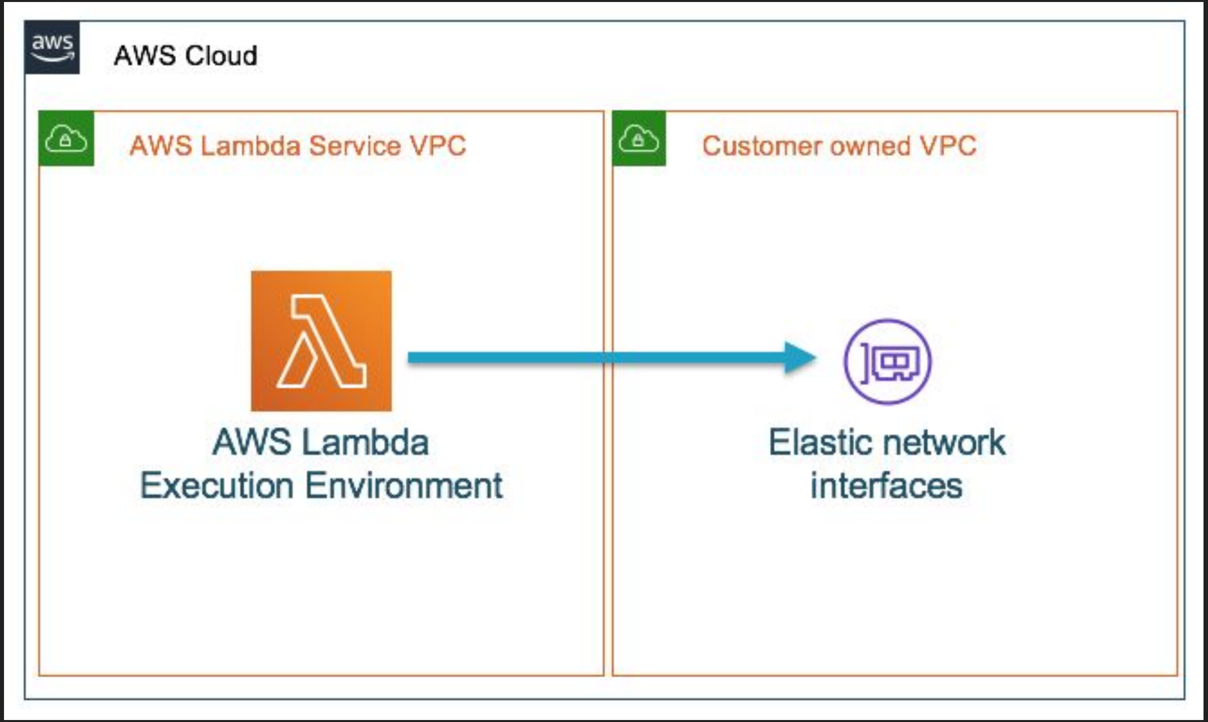
THE REQUEST LIFECYCLE



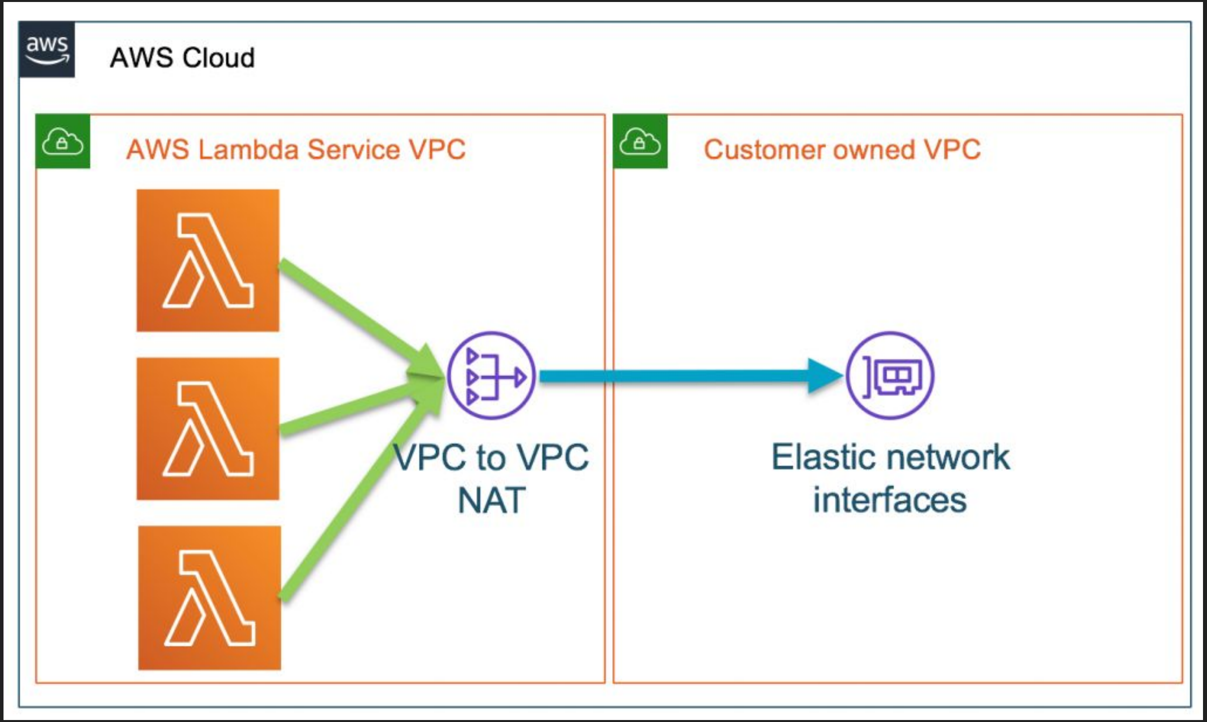
Reducing cold starts

- Bundle and minify your code
- Uninstall unnecessary dependencies
- Do not package aws-sdk - it’s already there

VPC cold starts - Before (up to 10 seconds)



VPC cold starts - After



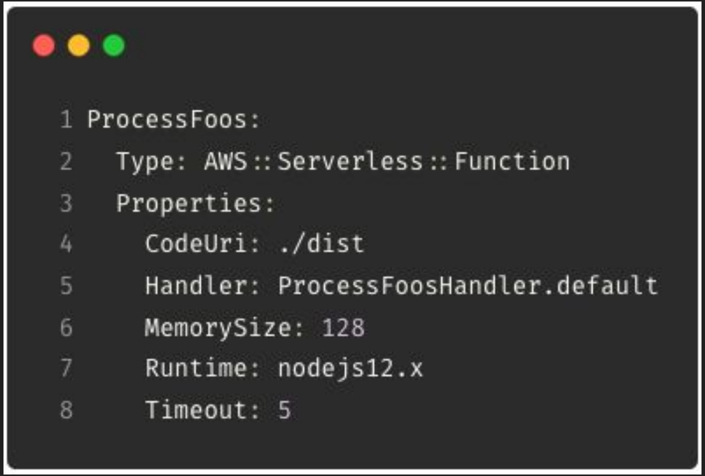
Database connections

Limit your lambda concurrency taking in consideration the number of your available connections.

IAC? SAM

Serverless Application Model is just an extension for CloudFormation.

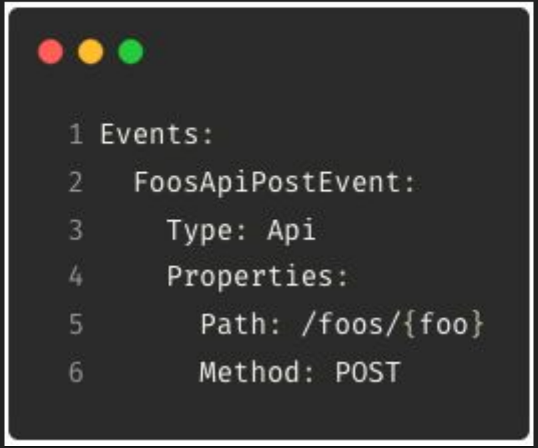
“Any resource that you can declare in an AWS CloudFormation template you can also declare in an AWS SAM template”



```
ProcessFoos:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ./dist
    Handler: ProcessFoosHandler.default
    MemorySize: 128
    Runtime: nodejs12.x
    Timeout: 5
```

SAM events

- Api
- SQS
- SNS
- S3
- DynamoDB
- Schedule
- Kinesis
- AlexaSkill
- Cognito
- IoTRule
- CloudWatchEvent
- CloudWatchLogs

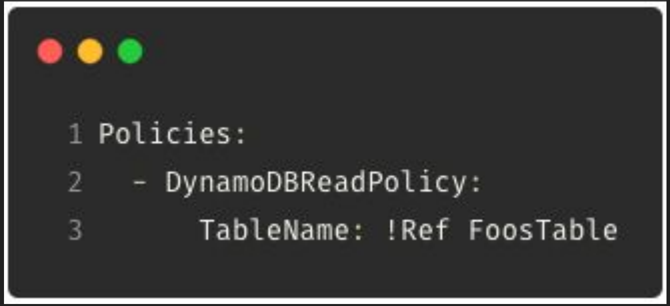


```
Events:
  FoosApiPostEvent:
    Type: Api
    Properties:
      Path: /foos/{foo}
      Method: POST
```

SAM policies

Follow least privilege principle:

- Specify only the needed services
- Use Read when there is no need for write permissions (crud)
- Boil down to specific resources

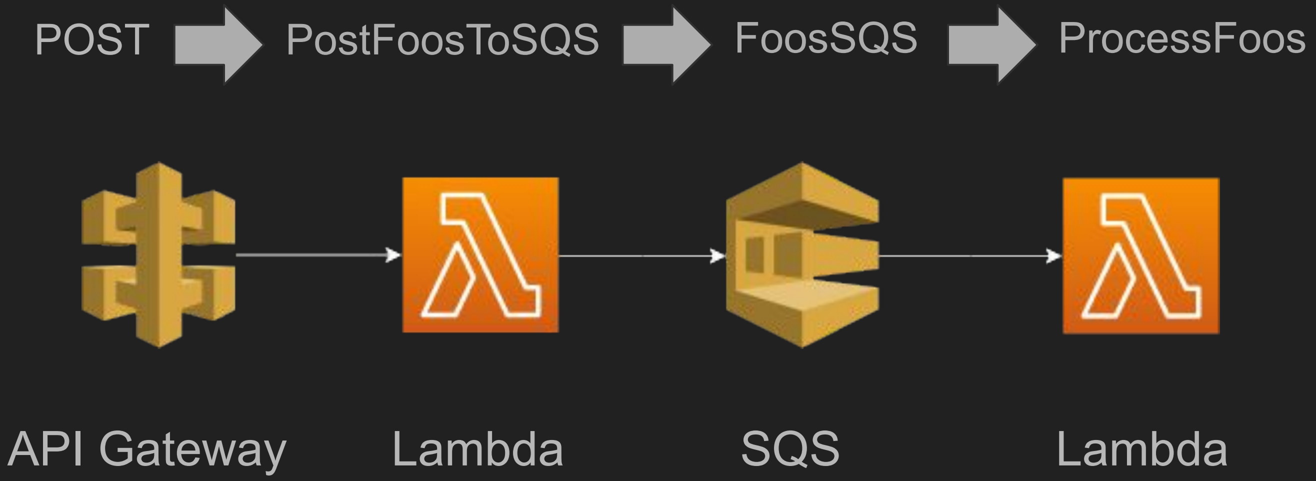


```yaml
Policies:
  - DynamoDBReadPolicy:
      TableName: !Ref FoosTable
```

SAM cli

- Trigger functions with any event locally
- Run api locally
- Package app
- Deploy app

Demo time


POST
PostFoosToSQS
FoosSQS
ProcessFoos
API Gateway
Lambda
SQS
Lambda

Wrap up

Some resources

- https://github.com/awslabs/serverless-application-model/blob/master/versions/2016-10-31.md
- https://d1.awsstatic.com/whitepapers/Overview-AWS-Lambda-Security.pdf
- https://d1.awsstatic.com/whitepapers/serverless-architectures-with-aws-lambda.pdf
- https://aws.amazon.com/blogs/compute/announcing-improved-vpc-networking-for-aws-lambda-functions/
- https://www.youtube.com/watch?v=QdzV04T_kec

Thank you

MINDERA

software craft