USE CASES FOR SERVERLESS TECHNOLOGIES

NICE TO MEET YOU

https://bit.ly/2Ul3L33

Ruslan Tolstov
Ruby Developer at MLSDev

AGENDA

- What is serverless?
- How serverless works?
- Providers
- AWS Lambda
- AWS API Gateway
- Serverless Frameworks
- Cons
- Prices
- Use cases

WHAT IS SERVERLESS?

Serverless architecture replaces long-running virtual machines with ephemeral compute power that comes into existence on request and disappears immediately after use

TRADITIONAL ARCHITECTURE

TRADITIONAL ARCHITECTURE

- Set up server

TRADITIONAL ARCHITECTURE

- Set up server
- OS

TRADITIONAL ARCHITECTURE

- Set up server
- OS
- DB

TRADITIONAL ARCHITECTURE

- Set up server
- OS
- DB
- Hardware updates

TRADITIONAL ARCHITECTURE

- Set up server
- OS
- DB
- Hardware updates
- Soft updates

TRADITIONAL ARCHITECTURE

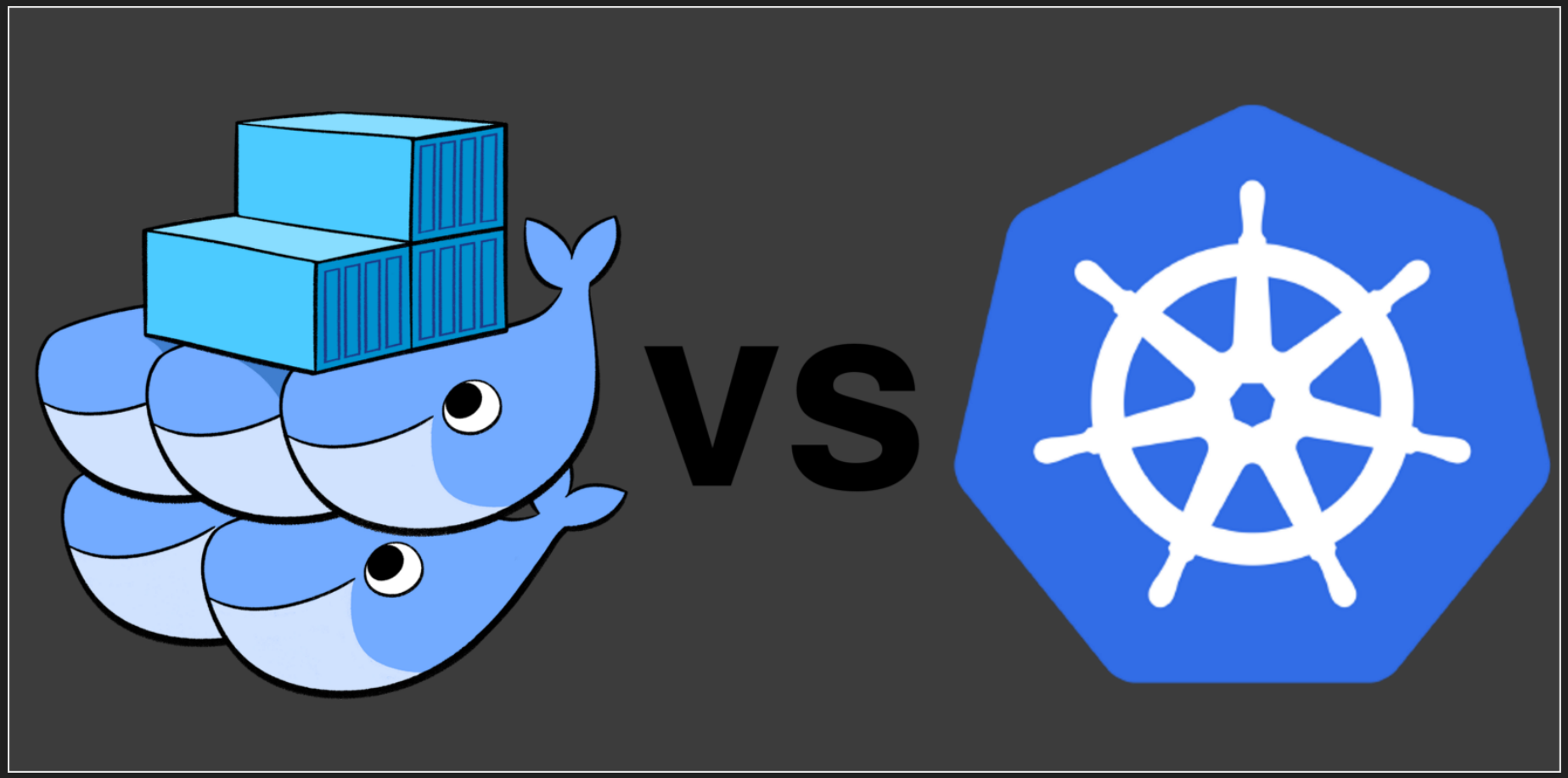
- Set up server
- OS
- DB
- Hardware updates
- Soft updates
- Scaling

TRADITIONAL ARCHITECTURE

- Set up server
- OS
- DB
- Hardware updates
- Soft updates
- Scaling

MICROSERVICE ARCHITECTURE

MICROSERVICE ARCHITECTURE



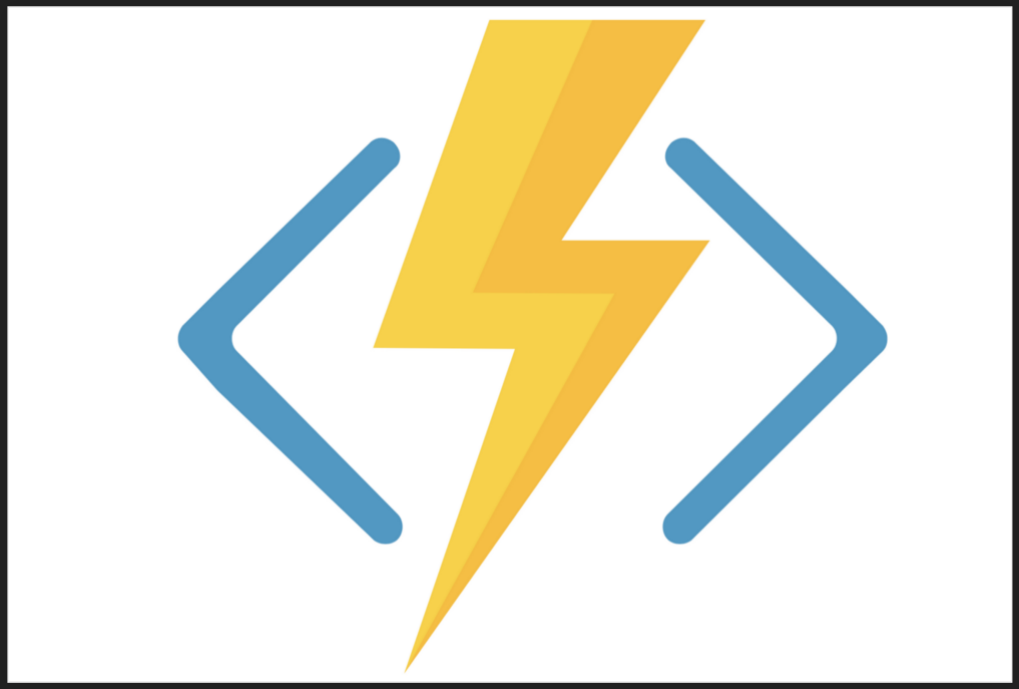
We need DevOps

SERVERLESS ARCHITECTURE

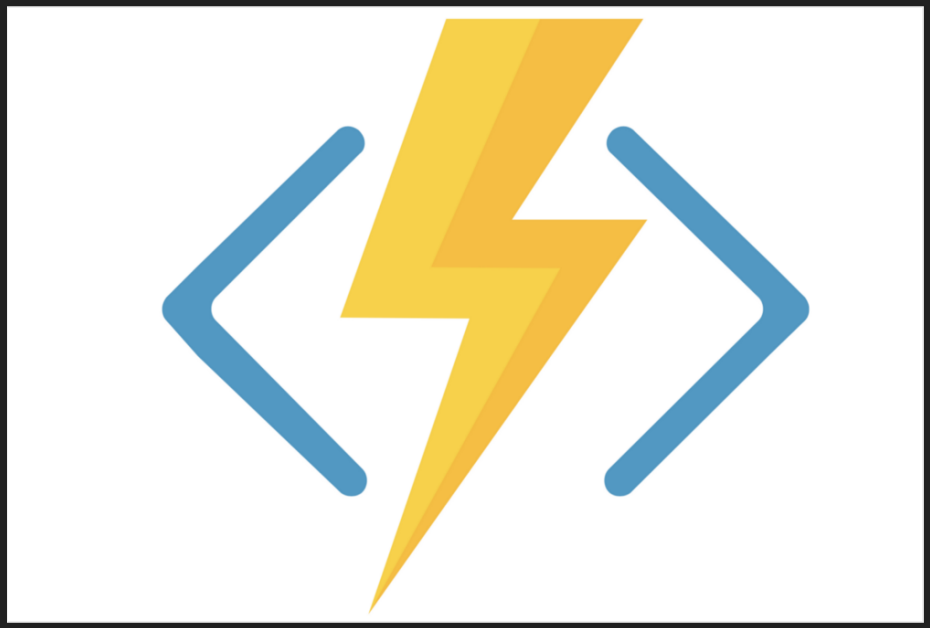
Function-as-a-Service(FaaS)

SERVERLESS ARCHITECTURE

Function-as-a-Service(FaaS)

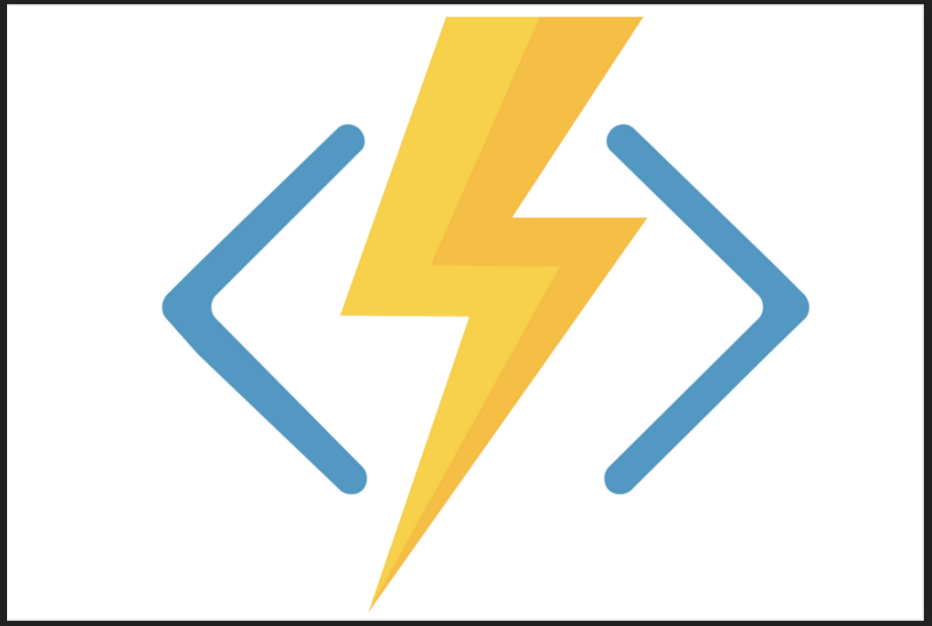


FUNCTION AS A SERVICE (FAAS)



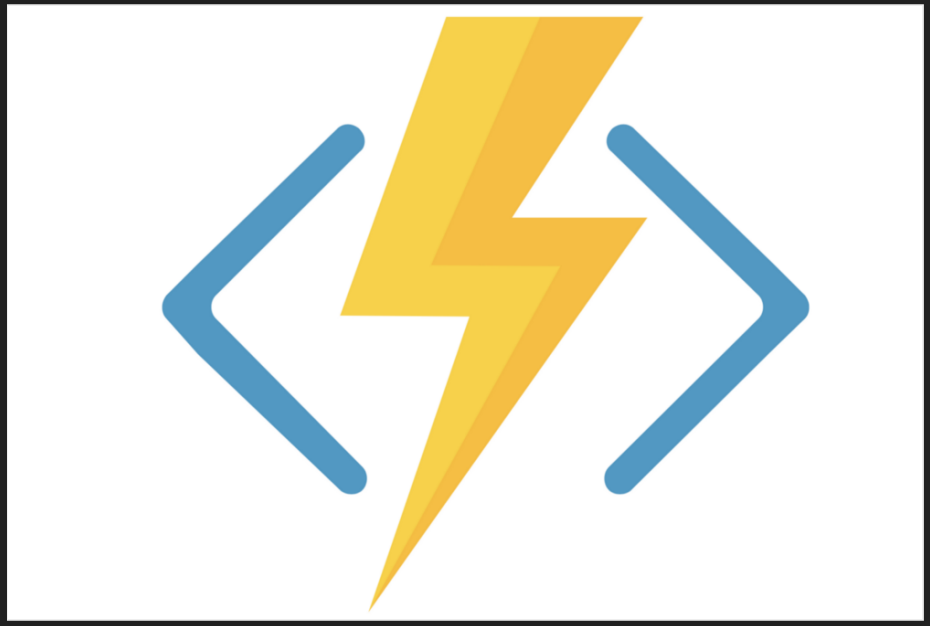
FUNCTION AS A SERVICE (FAAS)

- Function as the unit of deployment and scaling



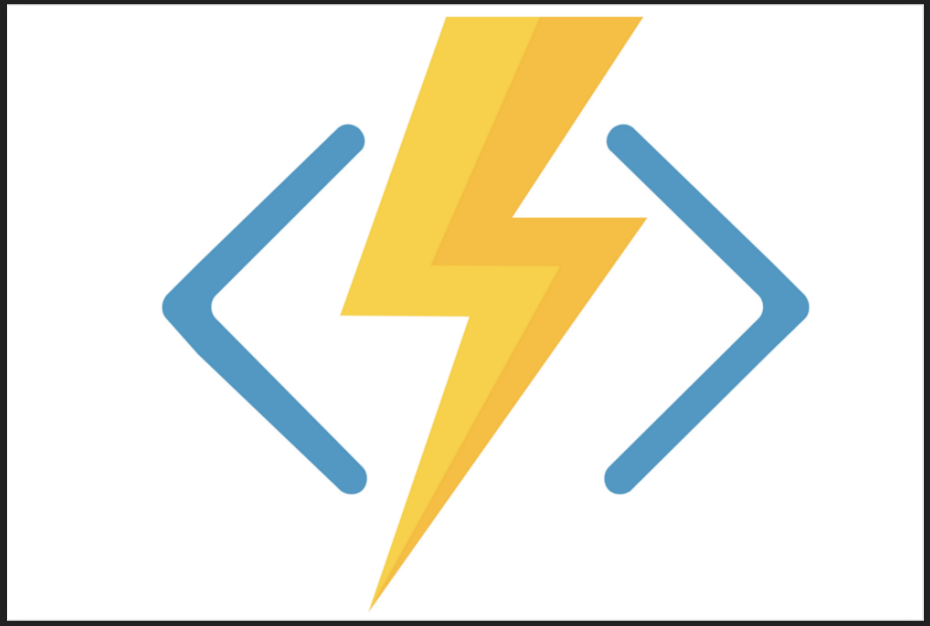
FUNCTION AS A SERVICE (FAAS)

- Function as the unit of deployment and scaling
- Metrics



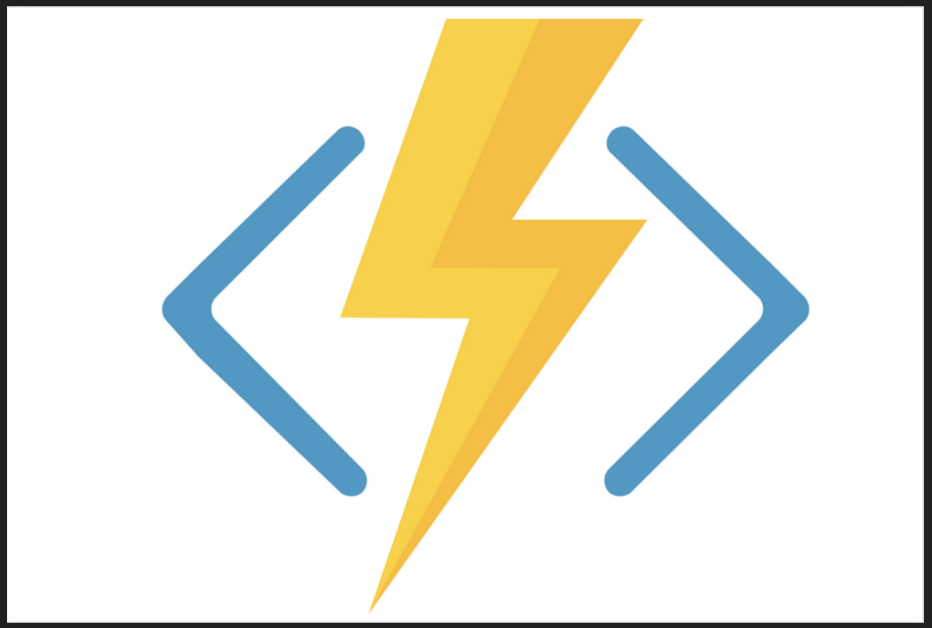
FUNCTION AS A SERVICE (FAAS)

- Function as the unit of deployment and scaling
- Metrics
- No machines, VMs, or containers



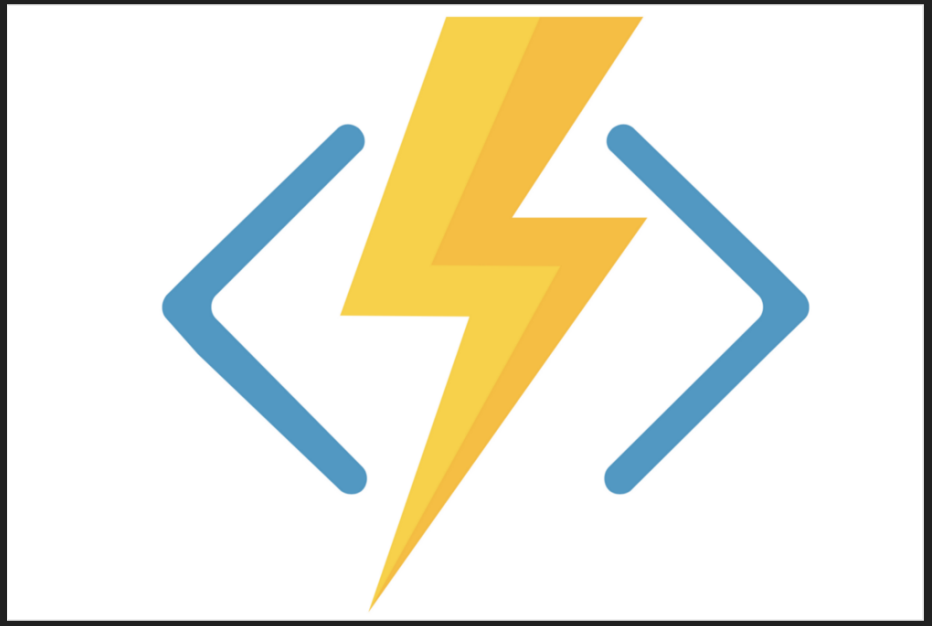
FUNCTION AS A SERVICE (FAAS)

- Function as the unit of deployment and scaling
- Metrics
- No machines, VMs, or containers
- Stateless

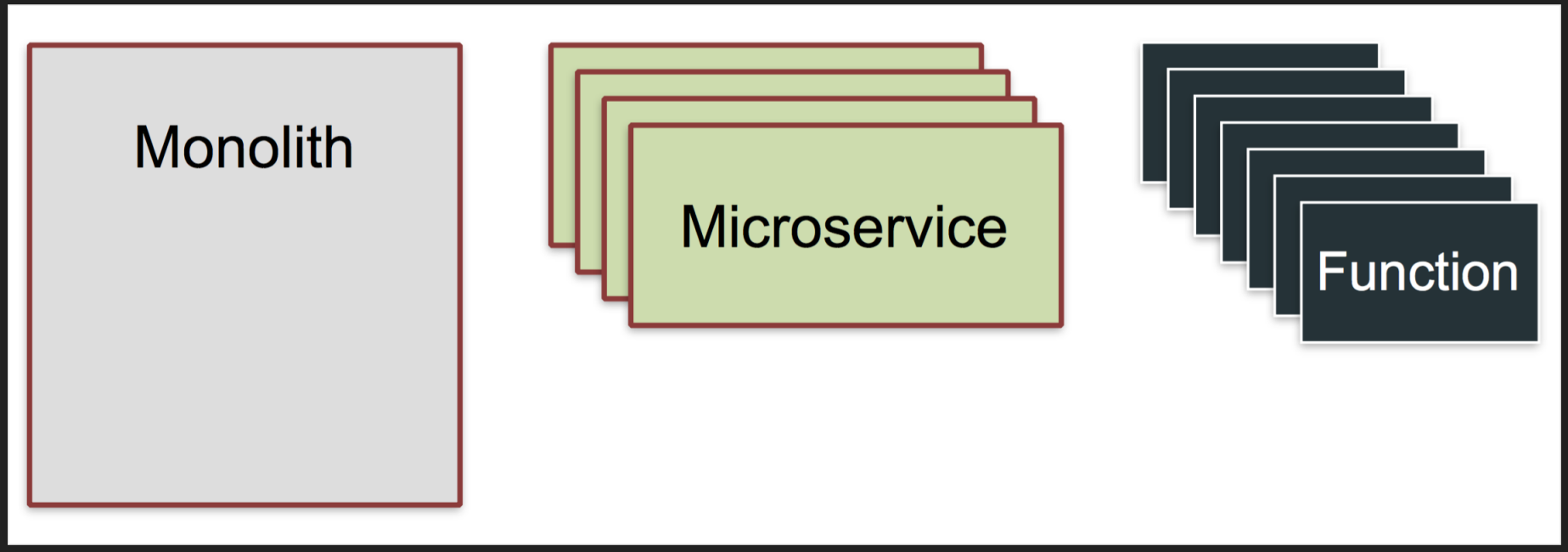


FUNCTION AS A SERVICE (FAAS)

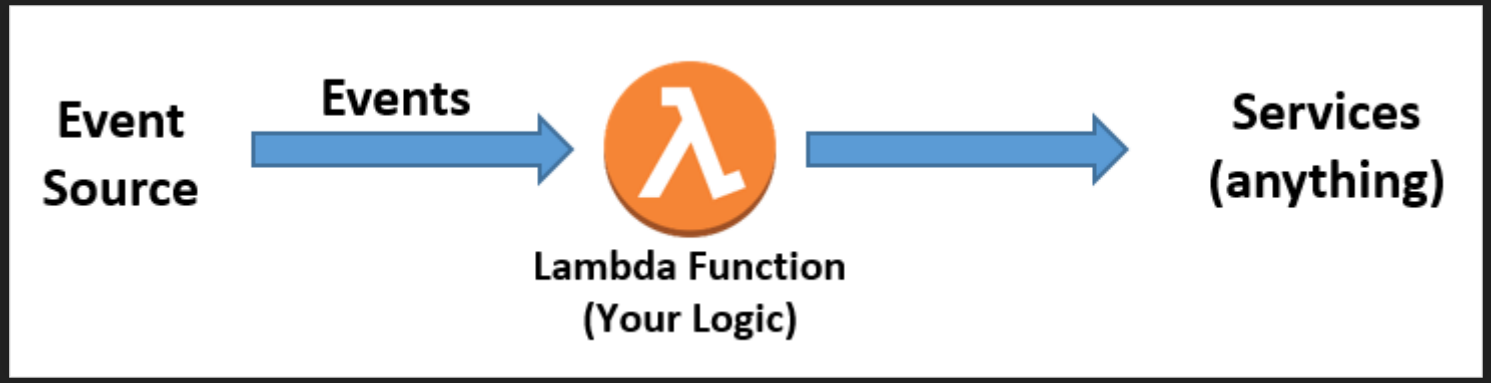
- Function as the unit of deployment and scaling
- Metrics
- No machines, VMs, or containers
- Stateless
- Never pay for idle



SERVERLESS EVOLUTION



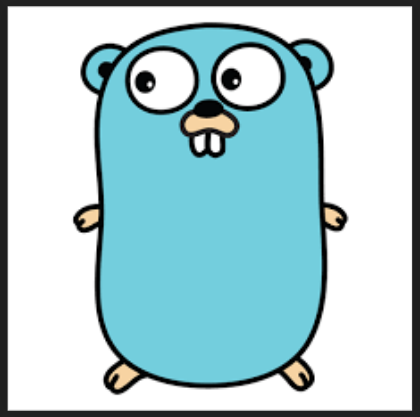
HOW SERVERLESS WORKS?



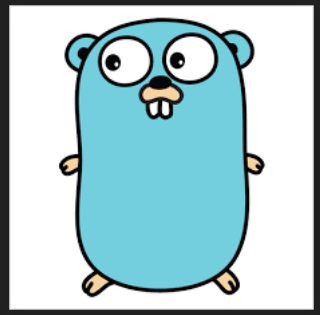
Functions are stateless

VENDORS

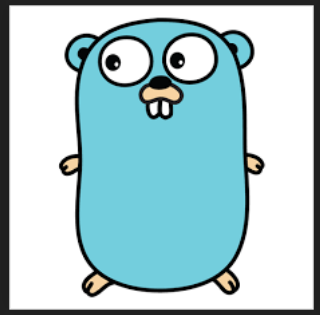
GCP SUPPORTED LANGUAGES



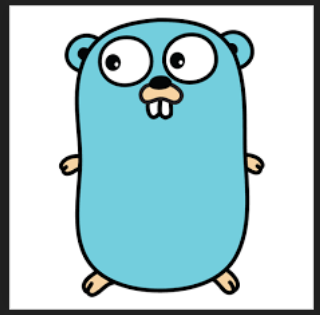
AWS SUPPORTED LANGUAGES

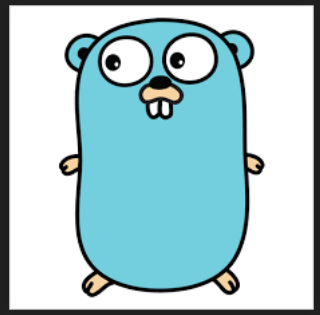

AWS SUPPORTED LANGUAGES

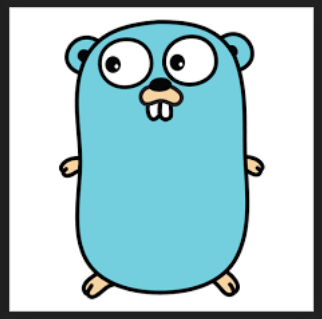

AWS SUPPORTED LANGUAGES

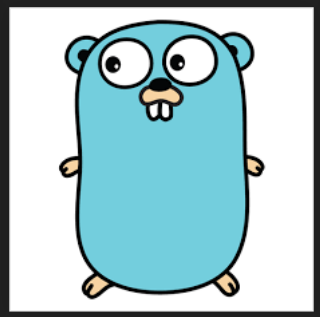

AWS SUPPORTED LANGUAGES

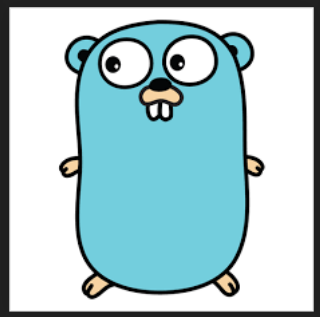

MS AZURE SUPPORTED LANGUAGES

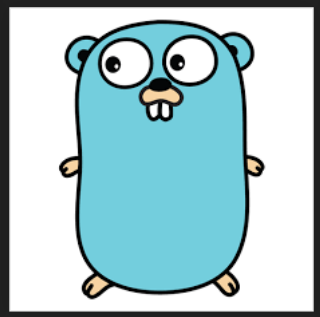

MS AZURE SUPPORTED LANGUAGES


MS AZURE SUPPORTED LANGUAGES

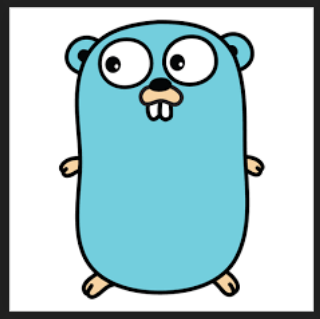




MS AZURE SUPPORTED LANGUAGES

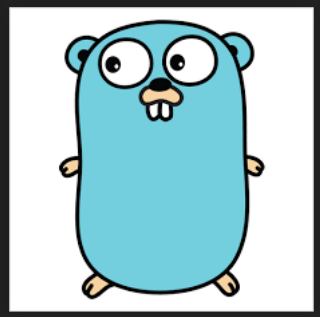


MS AZURE SUPPORTED LANGUAGES

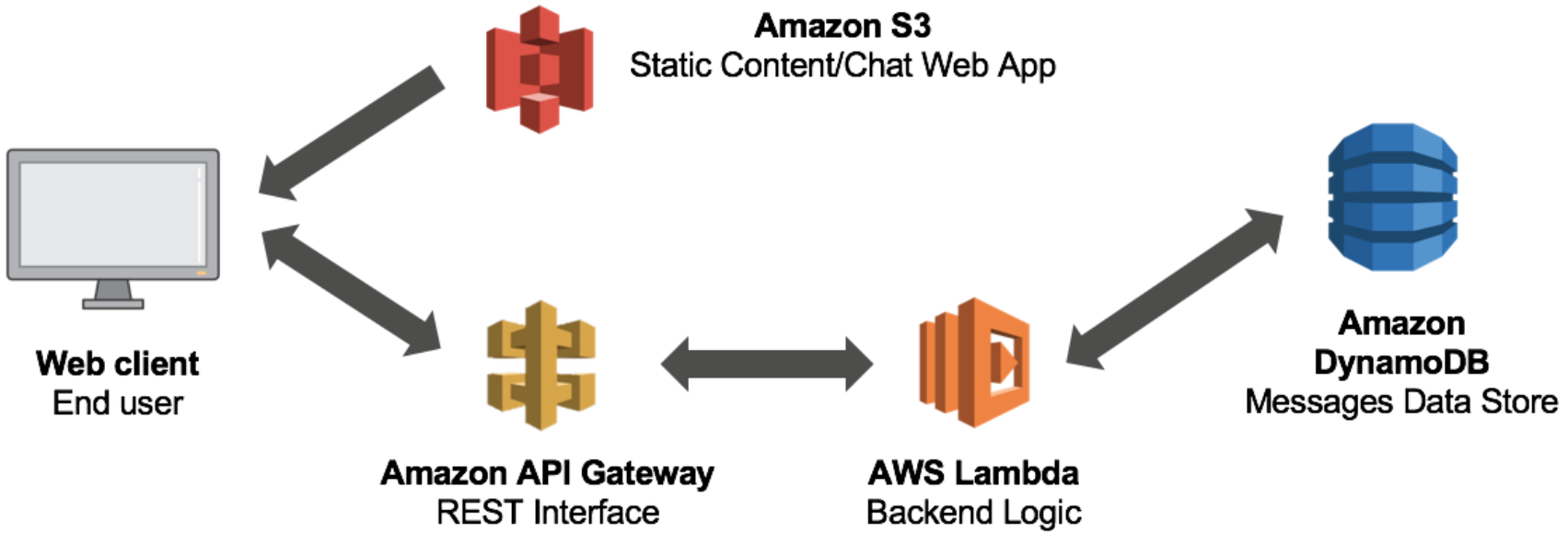


MS AZURE SUPPORTED LANGUAGES

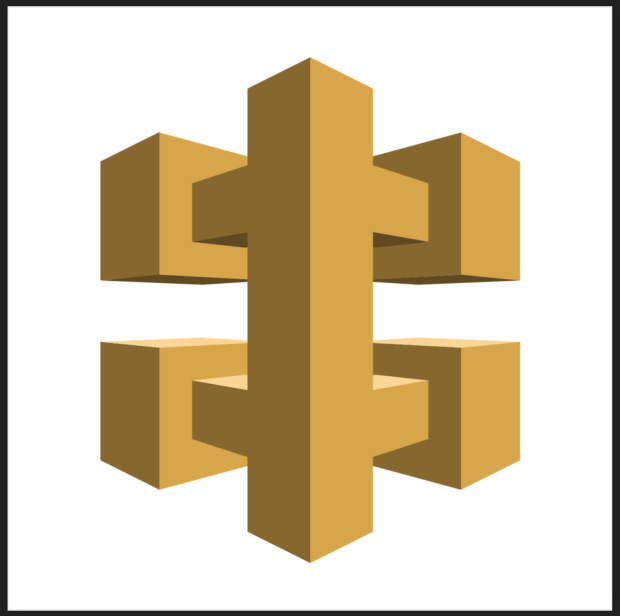






REST API EXAMPLE



AMAZON API GATEWAY

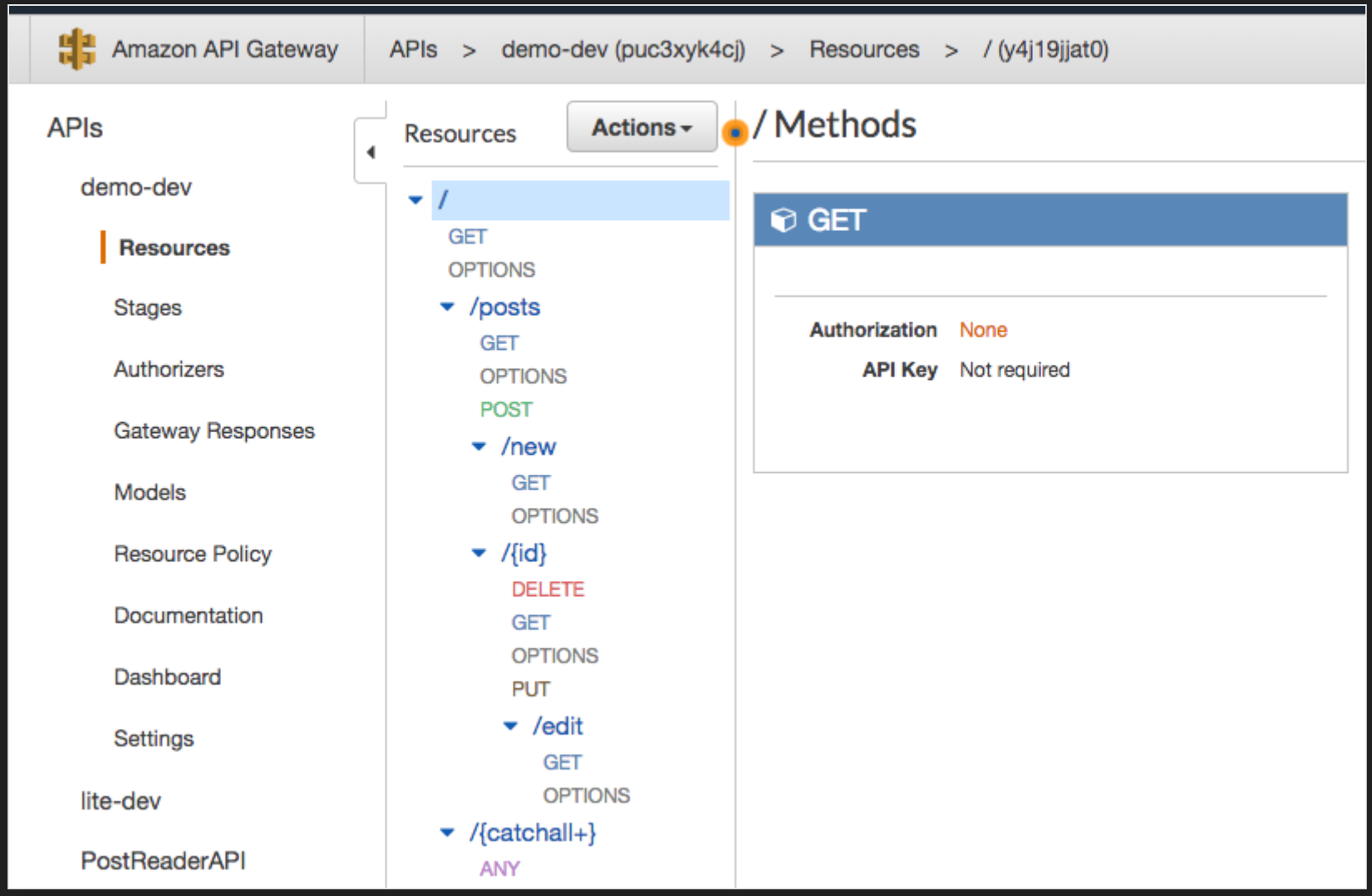


Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.

AMAZON API GATEWAY

- Easy API Creation and Deployment
- Support for HTTP/REST APIs and WebSocket APIs
- Resiliency
- API Lifecycle Management
- Monitoring
- Import from swagger/postman

ONLINE EDITOR API GATEWAY



AWS LAMBDA FUNCTION

- AWS Lambda is a compute service that lets you run code without procisioning or managing servers
- AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second
- Lambda functions can be used to join thogether many different AWS services
- AWS Services can trigger Lambda functions

ANATOMY OF LAMBDA FUNCTION (RUBY)

```ruby
def handler(event:, context:)
  #some logic
  { status: 200, body: {} }
end
```

ruby2.5

ANATOMY OF LAMBDA FUNCTION (NODEJS)

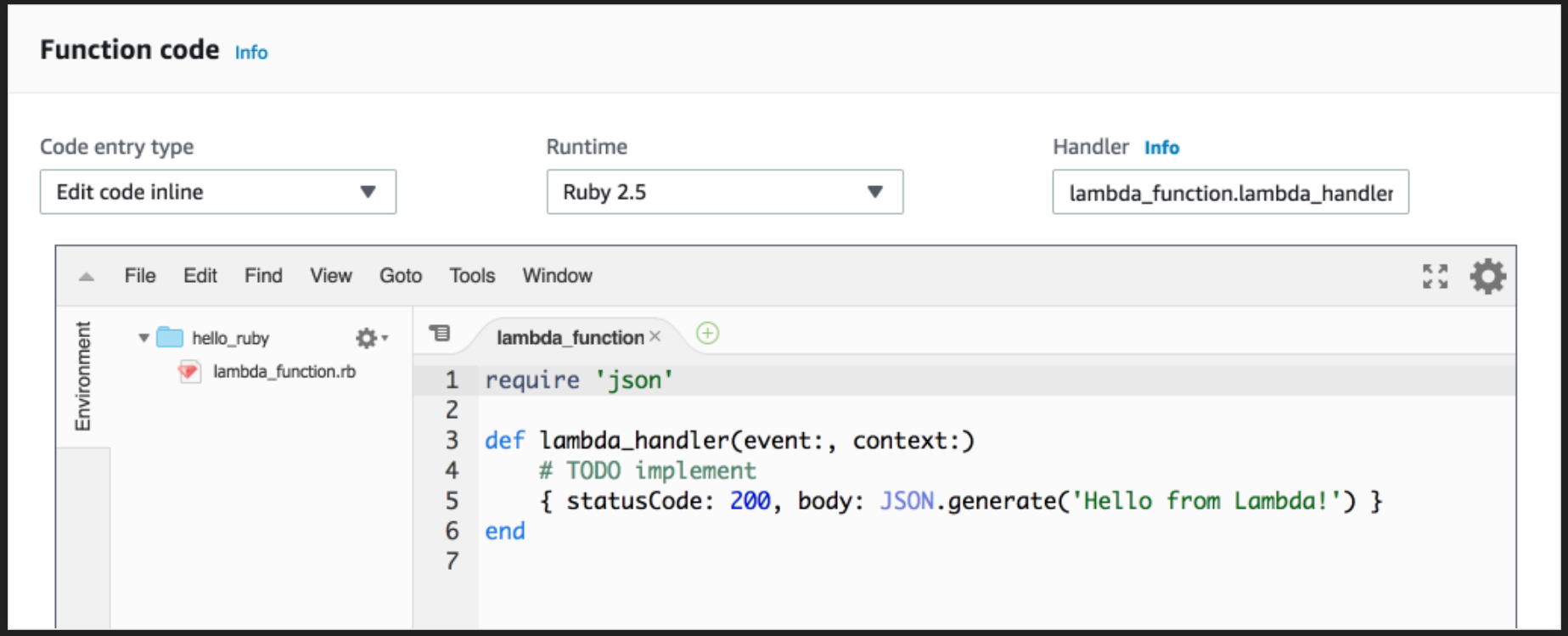
```
exports.handler = function(event, context, callback) {
 //some logic
 callback(null, { status: 200, body: {} })
}
```

nodejs6.10

```
exports.handler = async (event, context) => {
 //some logic
 return { status: 200, body: {} }
}
```

nodejs8.10

ONLINE EDITOR LAMBDA



AWS CLI + ZIP

```
cd lambda
zip –X –r ../index.zip *
aws lambda update-function-code --function-name
    MyLambdaFunction --zip-file fileb://index.zip
```

AWS CLI + CLOUDFORMATION OR SAM


```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "The AWS CloudFormation template for this Serverless application",
  "Resources": {
    "ResizeCategoryLogGroup": {
      "Type": "AWS::Logs::LogGroup",
      "Properties": {
        "LogGroupName": "/aws/lambda/lvg-develop-reports"
      }
    },
    "ResizeCategoryLambdaFunction": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "Code": {
          "S3Bucket": "lvg-cloudformation",
          "S3Key": "serverless/lvg/develop/1553255530306-2019-03-22T11:52:10.306Z/lvg.zip"
        },
        "FunctionName": "lvg-develop-reports",
        "Handler": "handler-reports.handler",
        "MemorySize": 320,
        "Role": "arn:aws:iam::704493629240:role/lvgRole",
        "Runtime": "nodejs8.10",
        "Timeout": 300,
        "Environment": {
          "Variables": {
            "STAGE": "develop",
```

SERVERLESS FRAMEWORKS

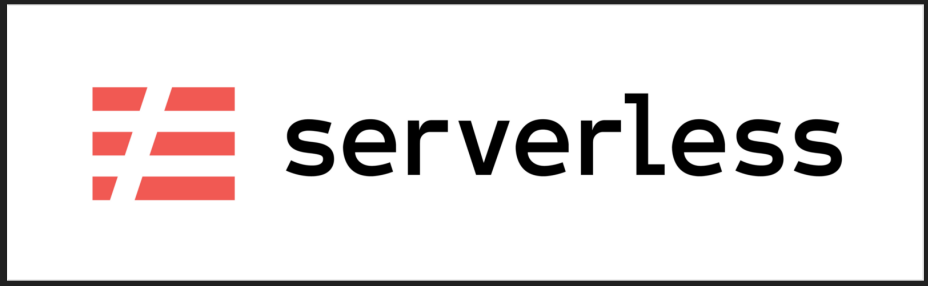
APEX
serverless architecture

Claudia.js

SPARTA

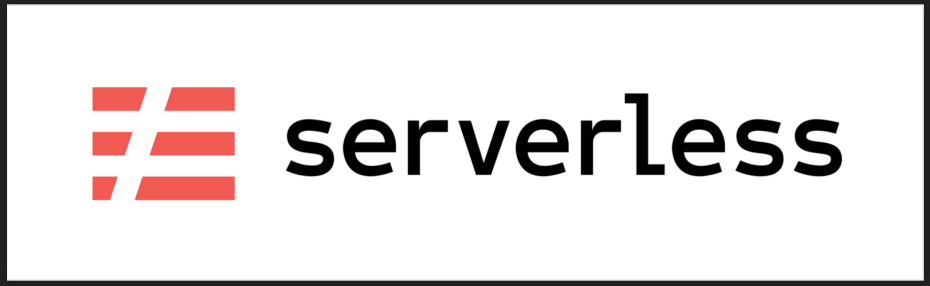
serverless

SERVERLESS FRAMEWORK



SERVERLESS FRAMEWORK

- Cross vendor support



SERVERLESS FRAMEWORK

- Cross vendor support
- Dozens of plugins

SERVERLESS FRAMEWORK

- Cross vendor support
- Dozens of plugins
- YAML

SERVERLESS FRAMEWORK

- Cross vendor support
- Dozens of plugins
- YAML
- Invoke function localy

SERVERLESS FRAMEWORK

- Cross vendor support
- Dozens of plugins
- YAML
- Invoke function localy
- MIT License

SERVERLESS CONFIG

```
service: serverless-thumbnail
runtime: ruby2.5
provider:
  name: aws
  timeout: 30
  memorySize: 128
  role: ${env:AWS_ROLE}
  deploymentBucket:
    name: ${env:DEPLOYMENT_BUCKET}
  iamRoleStatements:
  - Effect: Allow
    Action:
    - s3:putObject
    Resource: '*'
functions:
  thumbnail:
    events:
      - http: get thumbnail
    handler: handler.thumbnail
resources:
  Resources:
    NewResource:
      Type: AWS::S3::Bucket
```

DEPLOY

```
serverless deploy
```

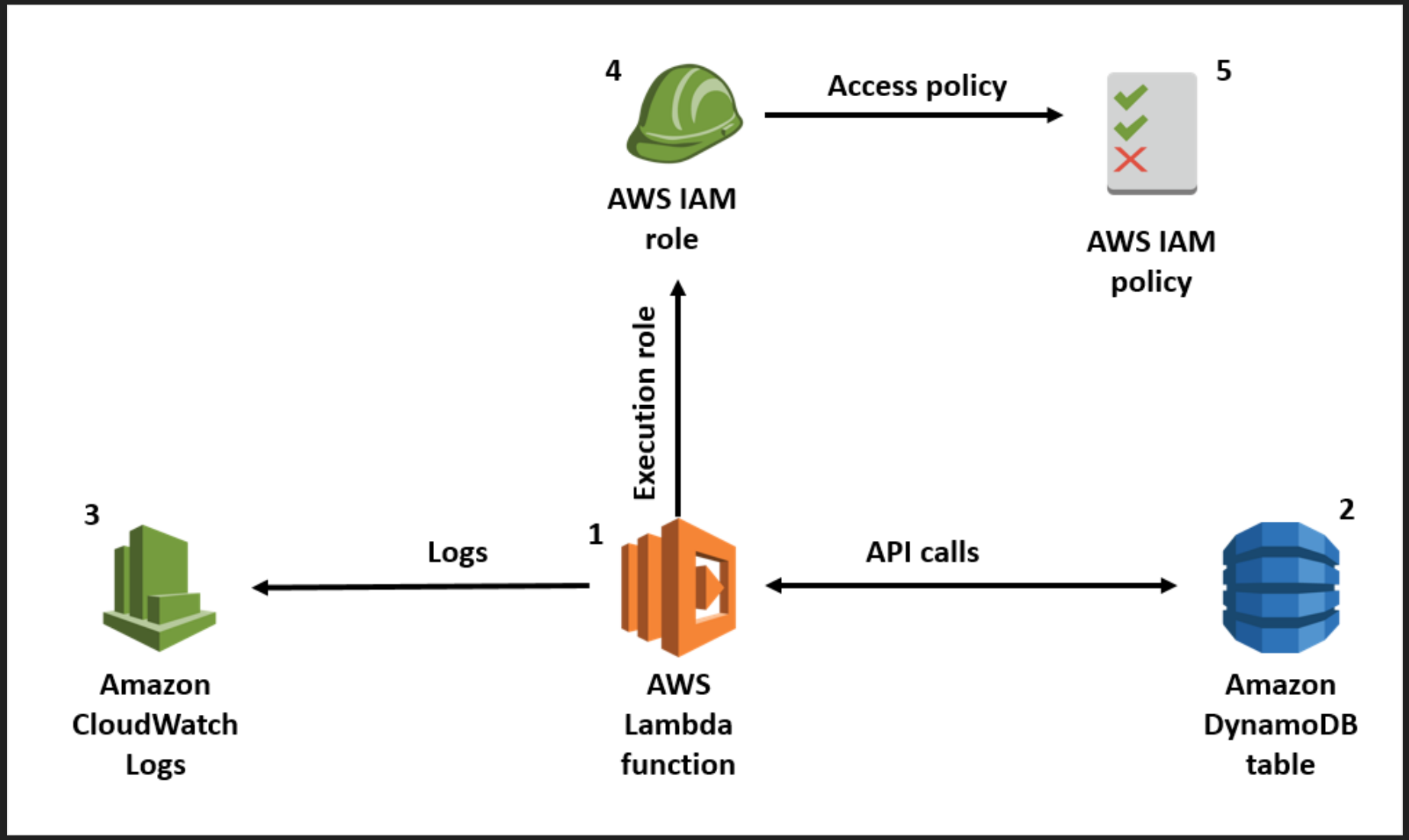
SERVERLESS CLI

```
serverless deploy -f sendEmail

serverless invoke local --function sendEmail

serverless logs -f sendEmail
```

SERVERLESS PLUGINS

- serverless-plugin-existing-s3
- serverless-dotenv-plugin
- serverless-secrets-plugin
- serverless-offline
- serverless-dynamodb-local

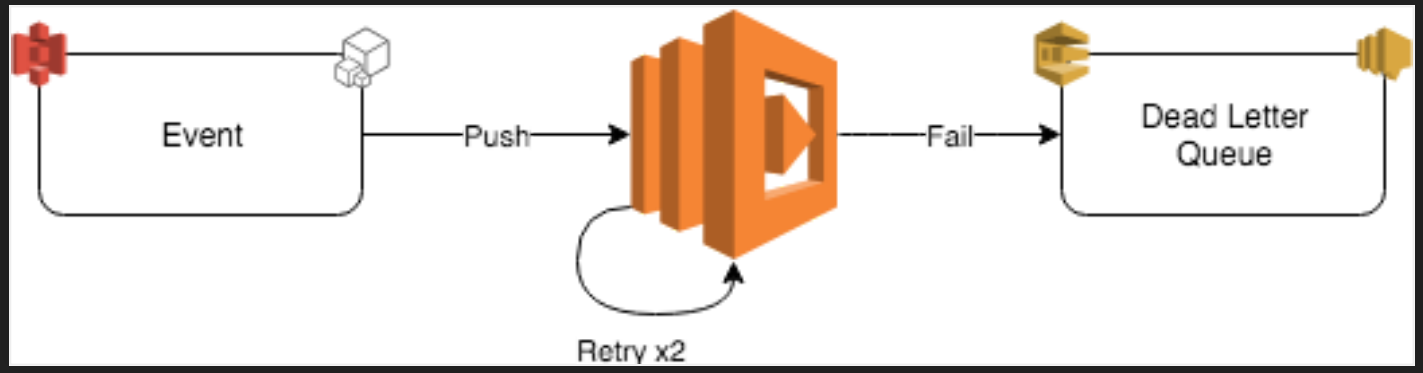
PERMISSIONS



INVOCATION TYPES

- Synchronous
- Asynchronous

RETRY BEHAVIOUR



DLQ - Dead Letter Queue

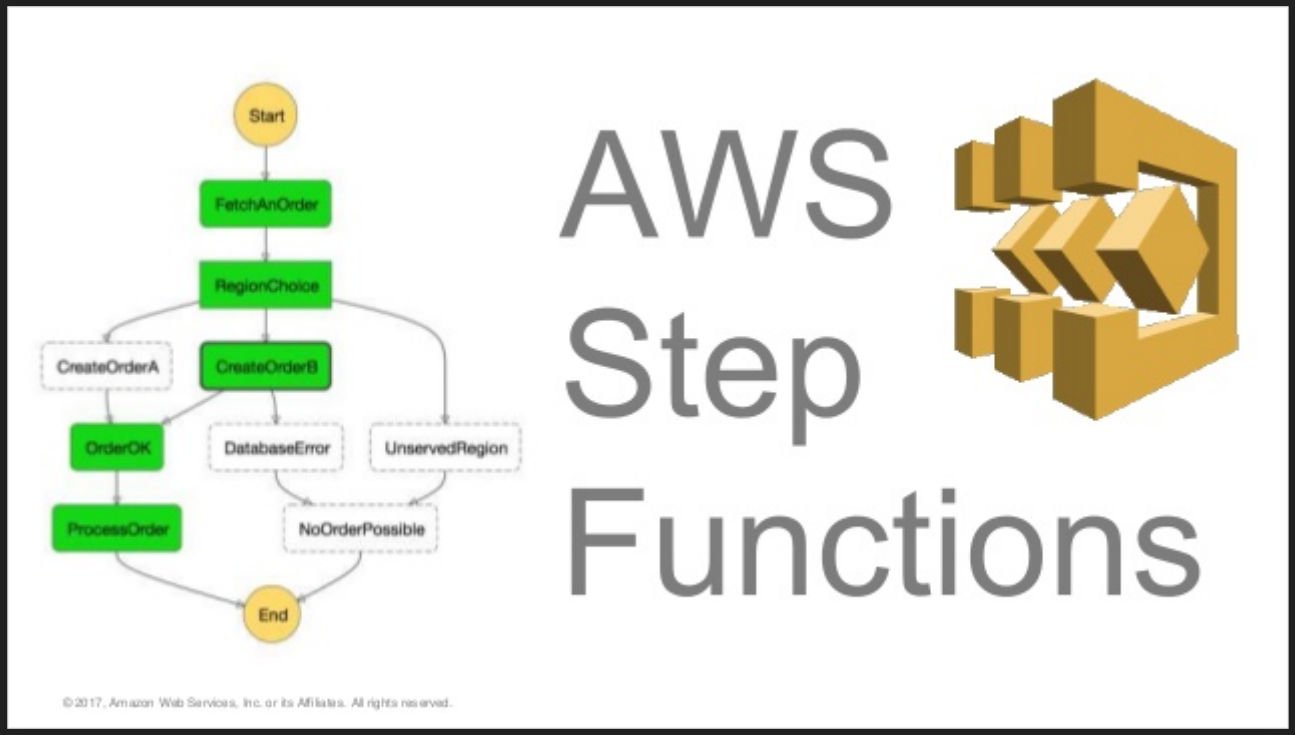
BINARIES

Your libraries must be for the Linux x64 platform/architecture.

```
npm --arch=x64 --platform=linux
```

or deploy with CI, docker container

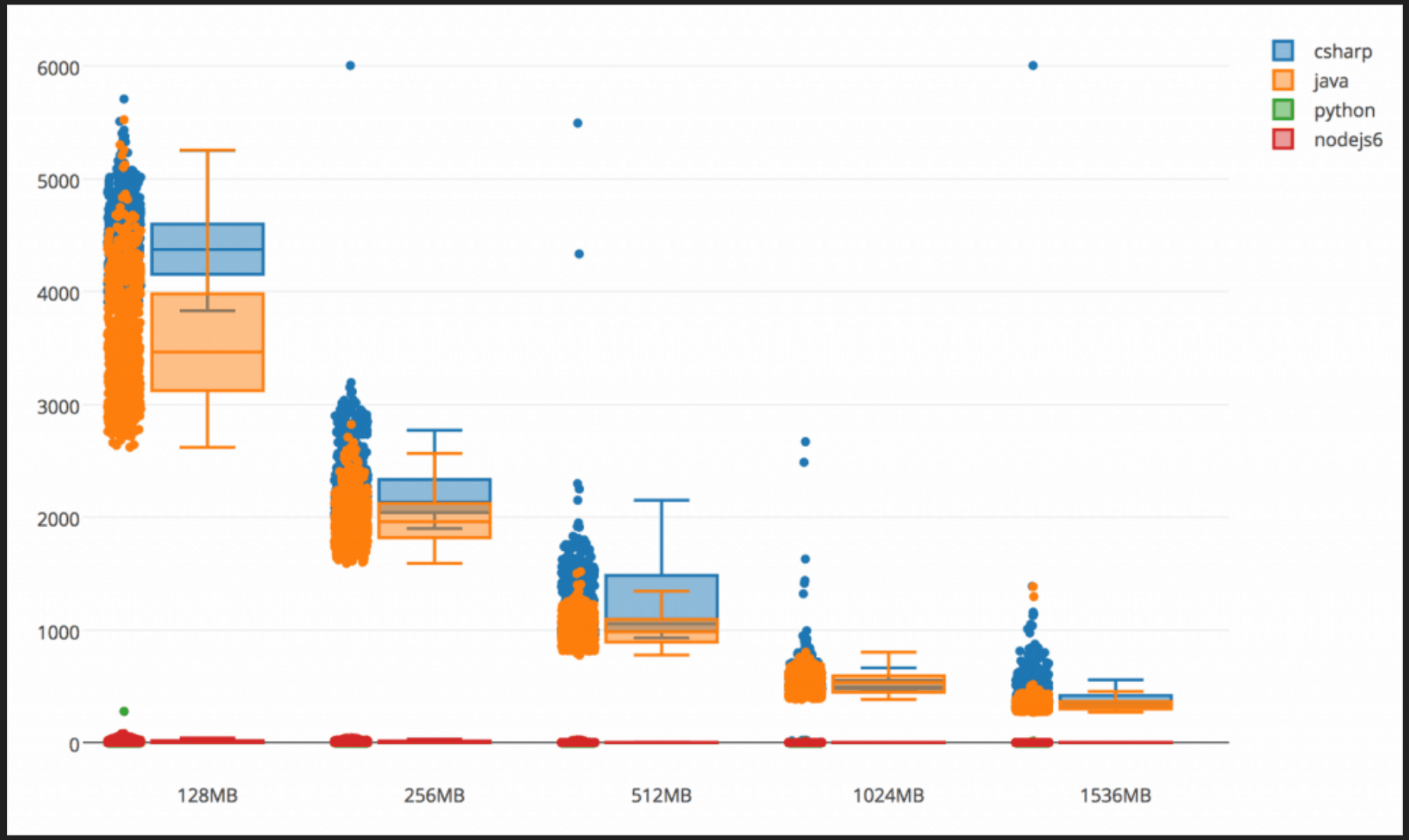
STEP FUNCTIONS



CONS

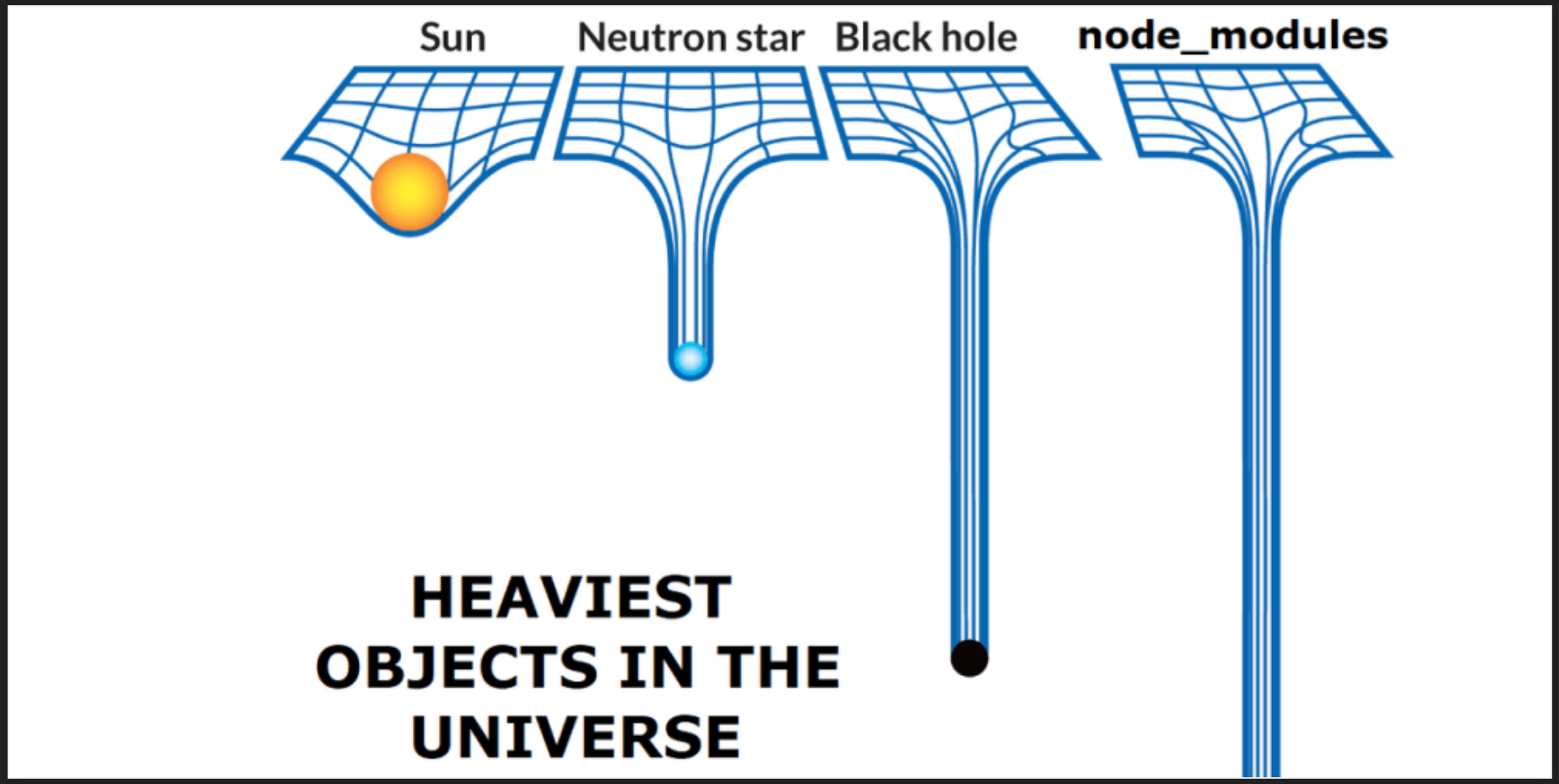
STATELESS

COLDSTART TIME

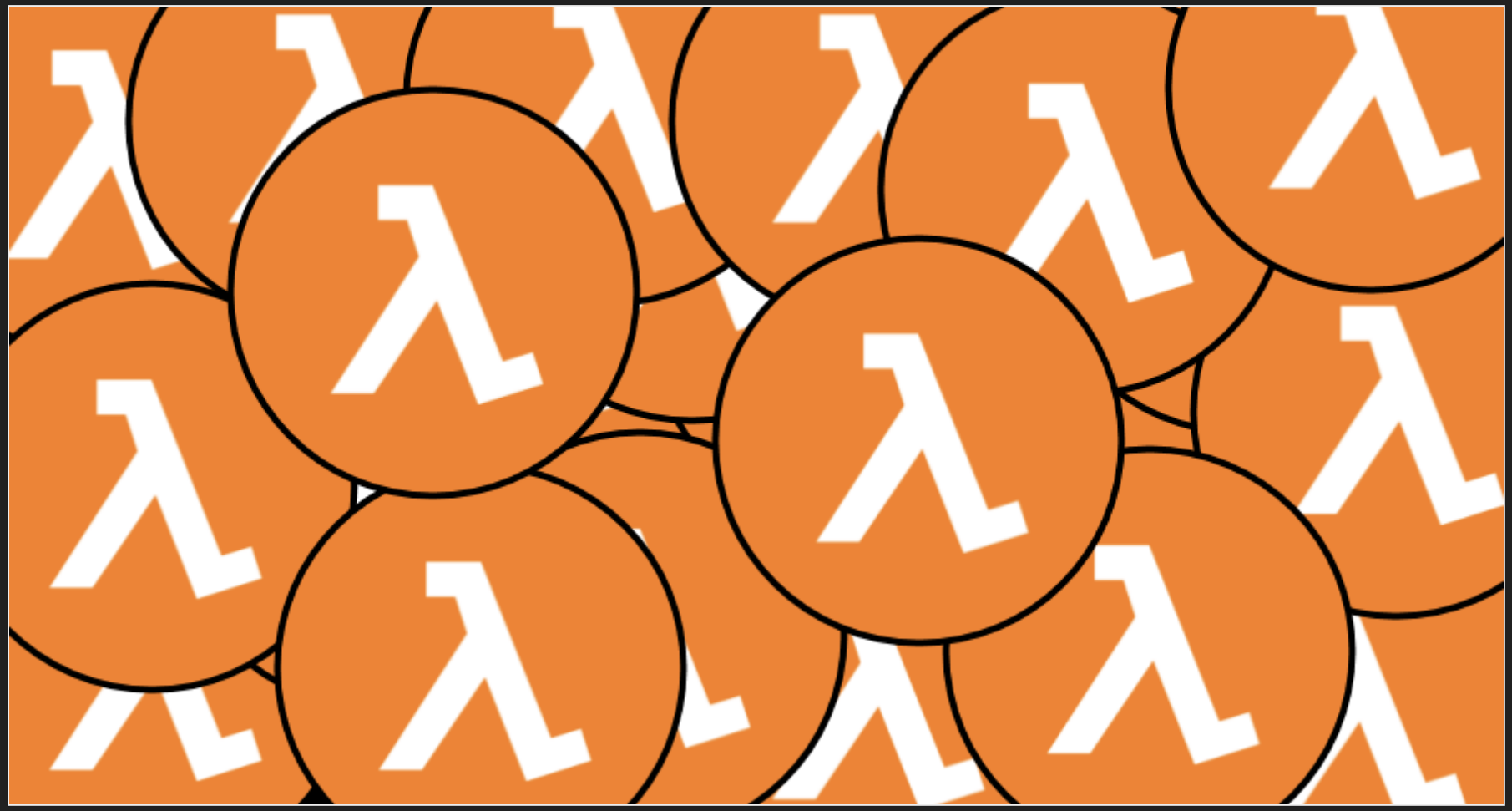


LAMBDA LIMITS

- /tmp directory storage - 512 MB
- Function timeout - 900 seconds (15 minutes)
- Body payload size - sync 6MB, async 128KB
- Function memory allocation - 128 MB to 3008 MB
- Deployment package size - 50 MB (zipped), 250 MB (unzipped)
- Concurrent executions - 1000 (per account)


Sun
Neutron star
Black hole
node_modules
HEAVIEST OBJECTS IN THE UNIVERSE

LAMBDA SCALING



1000+

VENDOR LOCK

OPENFAAS

OpenFaaS® (Functions as a Service) is a framework for building Serverless functions with Docker and Kubernetes which has first-class support for metrics. Any process can be packaged as a function enabling you to consume a range of web events without repetitive boiler-plate coding.

PRICES

FREE TIER

	IBM OpenWhisk	AWS Lambda	Azure Functions	GCP Functions
Requests	5M reqs	1M reqs	1M reqs	2M reqs
GB/s (compute time)	$0.000017 per GB-s	400,000 GB-s	400,000 GB-s	400,000 GB-s
Networking	None	None	None	5GB out

Compute Time = Invocations × Duration × RAM / 1024

PRICES

	IBM OpenWhisk	AWS Lambda	Azure Functions	GCP Functions
Requests	N/A	$0.2 per 1M requests	$0.2 per 1M requests	$0.4 Per 1M requests
GB/s (compute time)	$0.000017 per GB-s	$0.000016 per GB-s	$0.000016 per GB-s	$0.00000231/GB-s
Data Transfer (IN)	Free	$0.1-$0.2 between VPCs/regions	Free	Free
Data Transfer (OUT)	$0.05- $0.09 per GB	$0.05- $0.09 per GB	$0.05- $0.09 per GB	$0.12 per GB
API Gateway	Free	$3.50 per 1M calls	Free	$3.00 per 1M calls

LOW COMPUTE USE CASE

Low Compute Use Case

- Allocated memory 512 MB
- No. of invocations: 20,000 times/month
- Execution duration: 1 sec

AWS Lambda	
GB-sec = 20,000 * 512/1024	10,000 GB-sec
Compute charges = 10,000 * 0.00001667	$0.1667
Request charges = (20,000/1,000,000) * $0.2/Million	$0.004
Total	$0.1707

EC2 (on-demand t2.nano)	
$0.0081 * 24 * 30	$5.832
Total	$5.832

HIGH COMPUTE USE CASE

High Compute Use Case

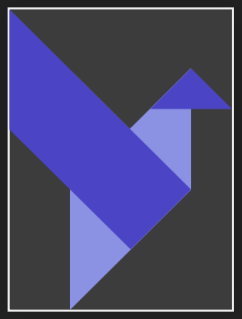
- Allocated memory 2496 MB
- No. of invocations: 30,000,000 times/month
- Execution duration: 500 sec

AWS Lambda	
GB-sec = 30,000,000 * 0.5s * 2496/1024	36,562,500 GB-sec
Compute charges = 36,562,500 * 0.00001667	$609.5
Request charges = (30,000,000/1,000,000) * $0.2/Million	$6
Total	$615.5

EC2 (on-demand m4.large)	
$0.192 * 24 * 30	$138.24
Total	$138.24

3RD PARTY TOOLS

- Epsagon
- Thundra
- Dashbird
- Sentry
- Rollbar



USE CASES

- Web appliactions
- IoT
- Media Transformation (audio/video/image)
- Multi-Location Media Transformation with **Lambda@Edge** and **CloudFront**
- Mass Emailing
- Real-time Data Transformation
- CRON Jobs with **CloudWatch**
- Chatbot
- ~~Websockets~~

CONCLUSIONS

- Keep the Lambda handler lean
- Avoid hardcoding, use enviroment variables
- One function, One task
- Watch the deployment package size, remove unused dependencies
- Make use of error handling mechanisme, DLQs
- Keep containers warm
- Use frameworks

THANKS!

QUESTIONS