

Serverless from the trenches

@Podgeypoos79 - @loige

loige.link/serverless-trenches

Padraig O'Brien



Head of Data and analytics



Connect

padraigobrien.com - [@podgeypoos79](https://twitter.com/podgeypoos79) - [padraigobrien](https://www.linkedin.com/in/padraigobrien) - [Linkedin](https://www.linkedin.com/company/planet9)



Luciano Mammino

**Fullstack
Bulletin**

fullstackbulletin.com



Principal Application Engineer



Connect

loige.co - [@loige](https://twitter.com/loige) - [lmammino](https://www.linkedin.com/in/lmammino) - [Linkedin](#)



Community Experience Distilled

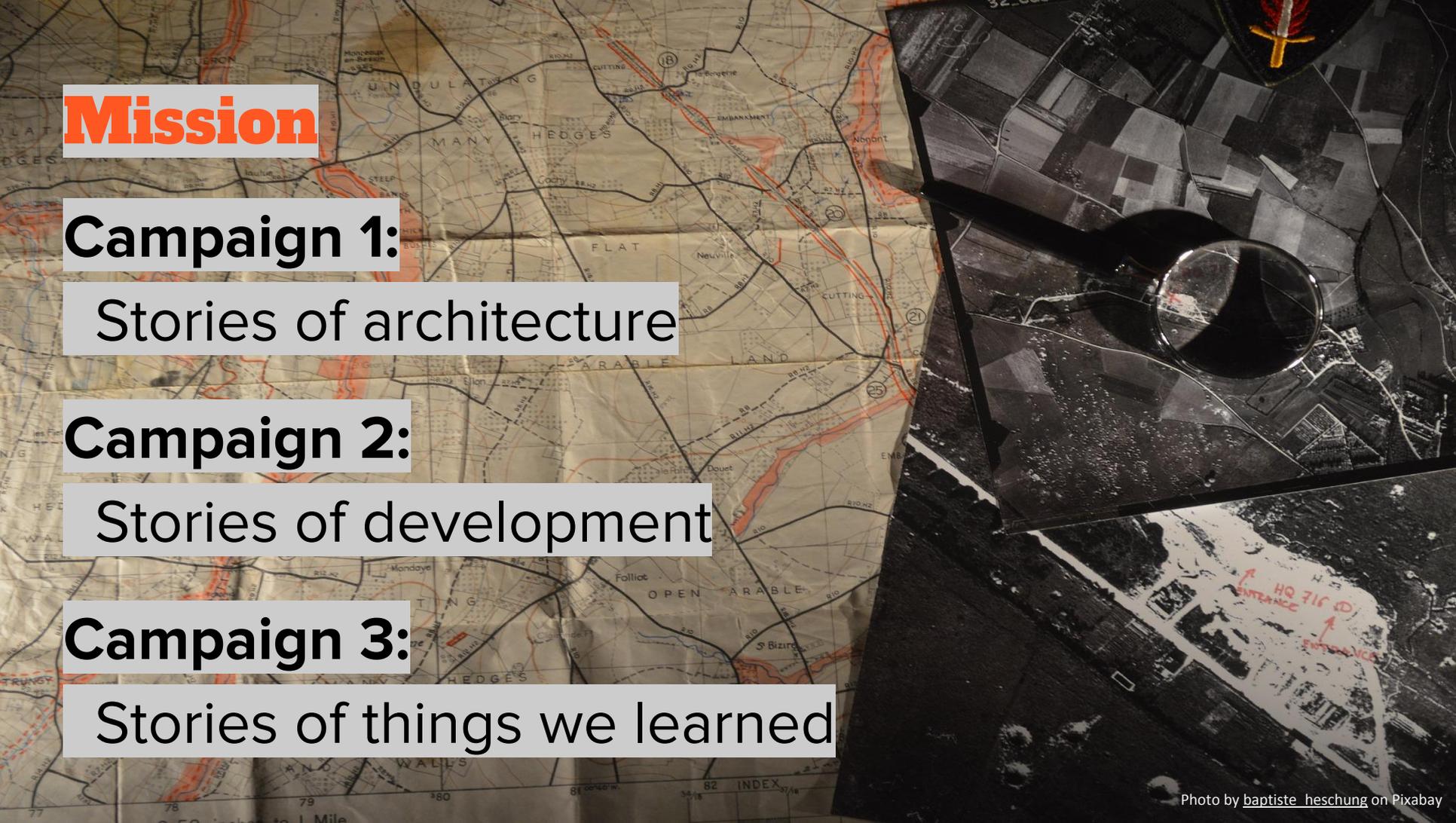
Node.js Design Patterns

Second Edition

Get the best out of Node.js by mastering its most powerful components and patterns to create modular and scalable applications with ease

Mario Casciaro
Luciano Mammino

[PACKT] open source*
PUBLISHING community experience distilled



Mission

Campaign 1:

Stories of architecture

Campaign 2:

Stories of development

Campaign 3:

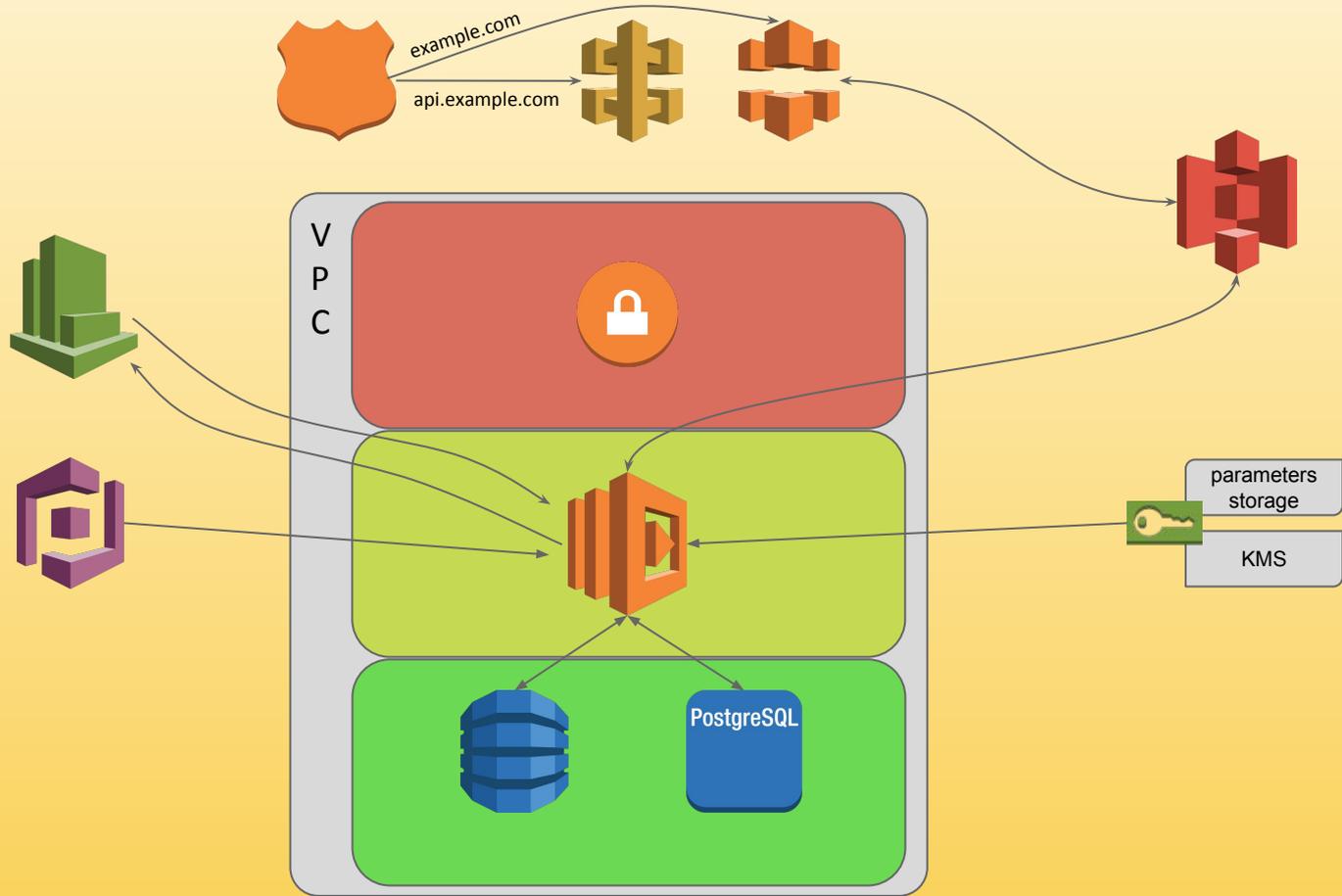
Stories of things we learned



Campaign 1.
Stories of architecture

There's more to it than API Gateway & Lambda



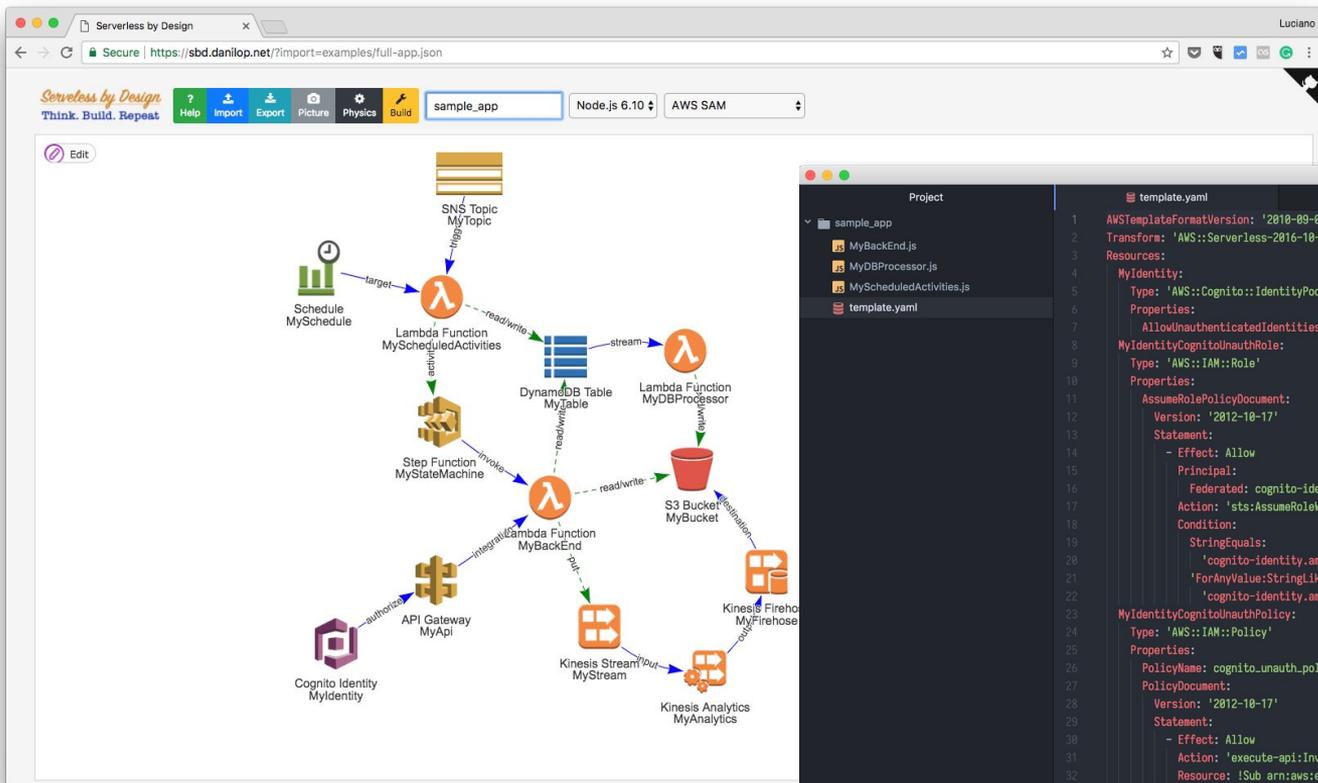


No shortcuts!

- **Spend a lot of time reading and learning!**
 - [Documentation](#) & [White papers](#)
 - [Security best practices](#)
- **If you like books:**
 - [Serverless architectures on AWS](#) (P. Sbarski)
 - [AWS Lambda in action](#) (D. Poccia)
- **"Infrastructure as code" is your friend:**
 - [Terraform](#) or [Cloudformation](#) & [SAM](#)
 - [Serverless framework](#)
- **If that's not enough... You can always ask for help :)**
 - [Version 1](#)
 - [Serverlesslab.com](#)



Serverless by design



```
Project
├── sample_app
│   ├── MyBackend.js
│   ├── MyDBProcessor.js
│   ├── MyScheduledActivities.js
│   └── template.yaml
└── template.yaml

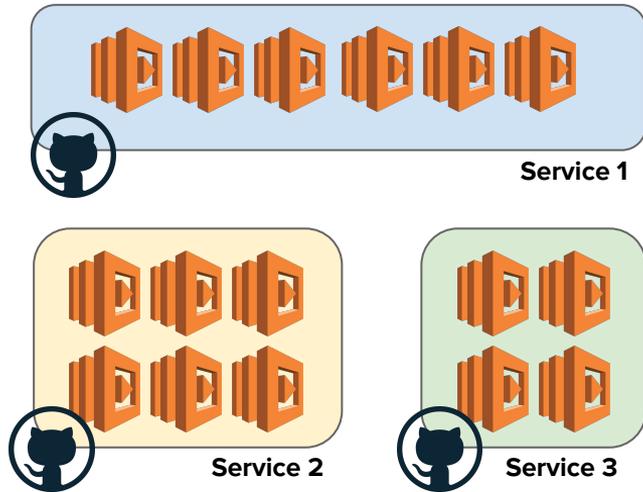
template.yaml
1  AWSTemplateFormatVersion: '2010-09-09'
2  Transform: 'AWS::Serverless-2016-10-31'
3  Resources:
4    MyIdentity:
5      Type: 'AWS::Cognito::IdentityPool'
6      Properties:
7        AllowUnauthenticatedIdentities: true
8    MyIdentityCognitoUnauthRole:
9      Type: 'AWS::IAM::Role'
10     Properties:
11       AssumeRolePolicyDocument:
12         Version: '2012-10-17'
13         Statement:
14           - Effect: Allow
15             Principal:
16               Federated: cognito-identity.amazonaws.com
17             Action: 'sts:AssumeRoleWithWebIdentity'
18             Condition:
19               StringEquals:
20                 'cognito-identity.amazonaws.com:aud': Ref! MyIdentity
21               'ForAnyValue:StringLike':
22                 'cognito-identity.amazonaws.com:amr': unauthenticated
23    MyIdentityCognitoUnauthPolicy:
24      Type: 'AWS::IAM::Policy'
25      Properties:
26        PolicyName: cognito_unauth_policy
27        PolicyDocument:
28          Version: '2012-10-17'
29          Statement:
30            - Effect: Allow
31              Action: 'execute-api:Invoke'
32              Resource: !Sub arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:*/*//*
33    Roles:
34      - !Ref MyIdentityCognitoUnauthRole
35    MyBackend:
36      Type: 'AWS::Serverless::Function'
37      Properties:
38        Handler: MyBackend.handler
39        Runtime: nodejs6.10
```

It's micro-services-ish



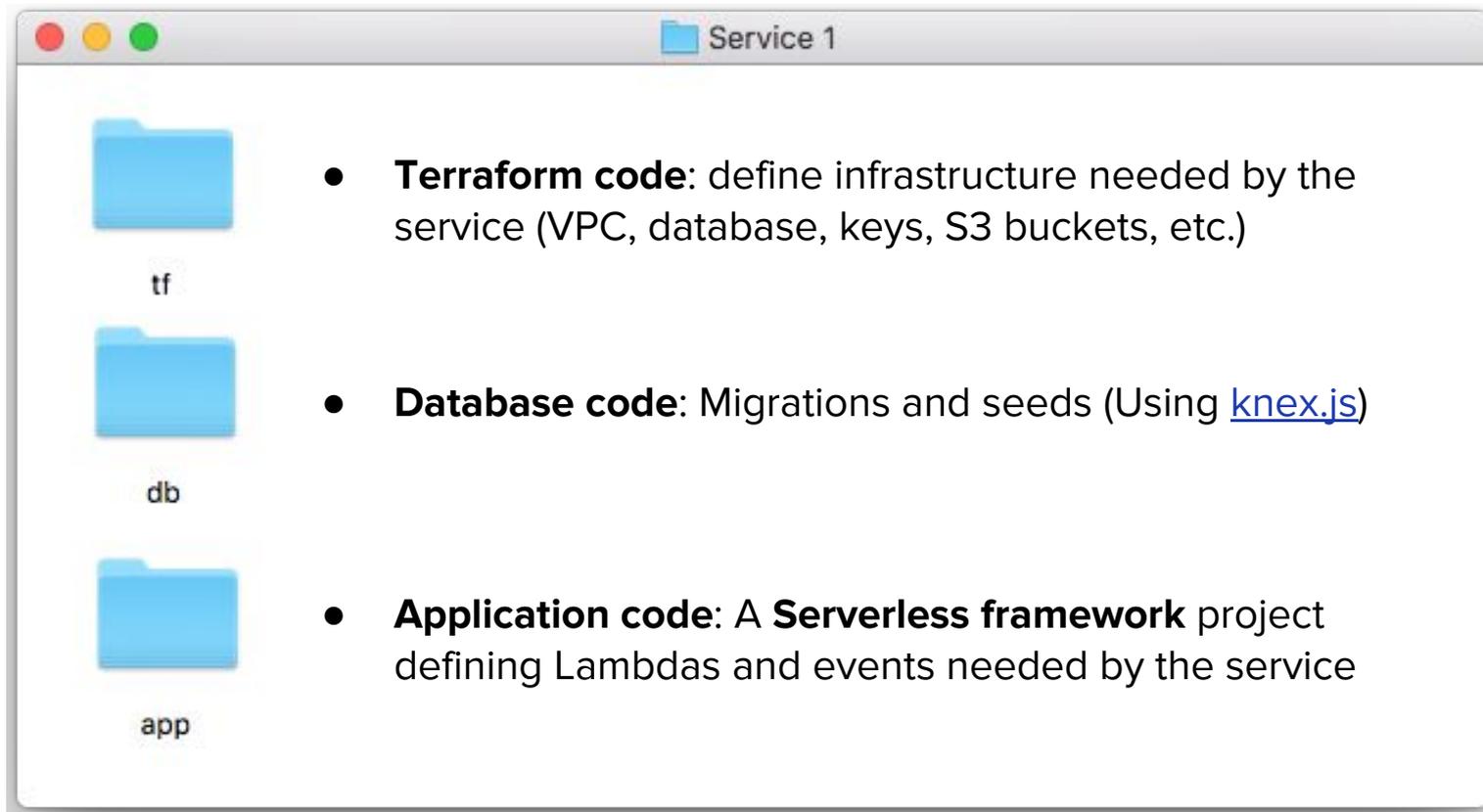
- A **function** (not a service) is the natural level of granularity!
- How to identify and structure services?
- How to connect services?
- How many repositories?
- How to deploy?
- Versioning?
- When and how to share code?

Functions are just building blocks!



- Proper service design using methodologies like Domain Driven Design.
- Find the bounded context of each service.
- Integration through message passing (events / APIs)
- Put everything related to a service into one repo.

How do we organise a service



The diagram illustrates the organization of a service into three distinct folders within a file explorer window titled "Service 1".

- **Terraform code:** define infrastructure needed by the service (VPC, database, keys, S3 buckets, etc.)
- **Database code:** Migrations and seeds (Using [knex.js](#))
- **Application code:** A **Serverless framework** project defining Lambdas and events needed by the service

When we integrate to master...

Our CI (Jenkins):

- Run tests
- Build the project
- Updates the infrastructure (Terraform)
- Updates the database (Knex)
- Deploy lambdas (Serverless framework)

(... of course we have multiple environments)



The anatomy of a typical Lambda function

```
(event, context, callback) => {
```

```
    // decrypt environment variables with KMS
```

```
    // deserialize the content of the event
```

```
    // validate input, authentication, authorization
```

```
    // REAL BUSINESS LOGIC (process input, generate output)
```

```
    // validate output
```

```
    // serialize response
```

```
    // handle errors
```

```
}
```

**BOILERPLATE
CODE**

**BOILERPLATE
CODE**



middy.js.org

The stylish Node.js middleware engine for AWS Lambda

```
const middy = require('middy')
const { middleware1, middleware2, middleware3 } = require('middy/middlewares')
```

```
const originalHandler = (event, context, callback) => {
  /* your pure business logic */
}
```

```
const handler = middy(originalHandler)
```

```
handler
```

```
.use(middleware1())
.use(middleware2())
.use(middleware3())
```

```
module.exports = { handler }
```

- **Business logic code is isolated:**

Easier to understand and test

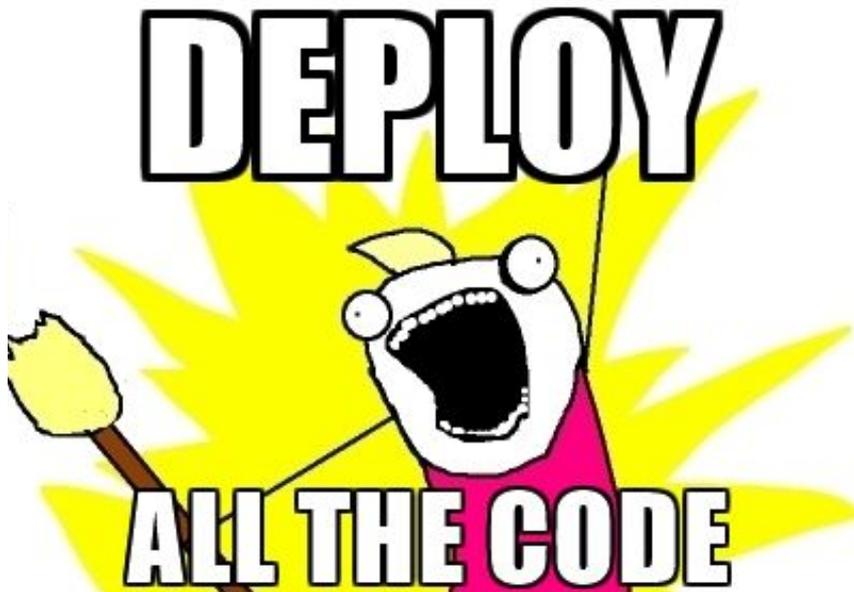
- **Boilerplate code is written as middlewares:**

- Reusable
- Testable
- Easier to keep it up to date



Campaign 2.
Stories of development

Organising development



- BDD
 - Discuss,
 - Distill
 - Acceptance criteria
- If you test, you [Jest](#)
- Commit to GitHub
(short lived *feature branches*)
- PR, Code Review & Merge
- Jenkins does the deploy magic.

Local development

- VS Code, Sublime, VIM.
- Postgres locally using docker.
- [Knex.js](#) to create the DB and seed.
- No local api gateway from AWS.
- Serverless, [serverless-webpack](#) and [serverless-offline](#) to simulate api gateway

```
Luciano ~  
| docker run --name some-postgres -e POSTGRES_PASSWORD=mysecretpassword -d postgres  
Unable to find image 'postgres:latest' locally  
latest: Pulling from library/postgres  
aa18ad1a0d33: Already exists  
436a6256db12: Pull complete  
6197c903e49f: Pull complete  
81e753951865: Pull complete  
f22fa190326d: Pull complete  
ee77fd7e88a3: Pull complete  
fc5111d03bdf: Pull complete  
1ca8a9c099f8: Pull complete  
8e2b99fddd9: Pull complete  
6f4ad1ab8ef2: Pull complete  
677839861264: Pull complete  
06d98194bfad: Pull complete  
Digest: sha256:317a541c064ec73cb48dedc2fbbb7797dd075303a2d9befcb85641e6ff6bf10f  
Status: Downloaded newer image for postgres:latest  
1e25197af9043636dcdcf387107069a0f69bf15c079815508ffc057e98e3eed3f  
Luciano ~
```

Serverless Application Model (SAM)



- Exploring [SAM](#) and [SAM Local](#)
- Uses similar markup to Serverless.
- SAM Local uses Docker.
- Step in the right direction for AWS in regard to local dev tooling



Campaign 3.

Stories of things we learned (the hard way...)

Large services

Template format error: Number of resources, 214, is greater than maximum allowed, 200

- Cloudformation has a hard limit
- Maximum of 200 resources
-
- [serverless-plugin-split-stacks](#)
 - migrates the RestApi resource to a nested stack



API Gateway & Lambda size limits

- 128 K payload for async event invocation
- 10 MB payload for response
- Don't find these limits when using [sls webpack serve](#)



API Gateways events

https://myapi.me?name=me



```
{  
  "requestContext": { ... },  
  "queryStringParameters": {  
    "name": "me"  
  },  
  "headers": { ... }  
}
```

```
const handler = (event, context, callback) {  
  console.log(event.queryStringParameters.name)  
  // ...  
}
```

It will output "me"

API Gateways events

https://myapi.me

(no query string!)



```
{  
  "requestContext": { ... },  
  (no queryStringParameters key!)  
  "headers": { ... }  
}
```

```
const handler = (event, context, callback) {  
  console.log(event.queryStringParameters.name)  
  // ...  
}
```

undefined

TypeError: Cannot read property 'name' of undefined

API Gateways events

```
const handler = (event, context, callback) {  
  if (event.queryStringParameters) {  
    console.log(event.queryStringParameters.name)  
  }  
  // or  
  console.log(event.queryStringParameters ?  
    event.queryStringParameters.name : undefined)  
}
```

**MOAR
boilerplate!**

[Api Gateway proxy event normalizer](#) middleware is coming to Middy!

API Gateways custom domain.

- Serverless does not provide custom domain name mapping
- Has to be done in cloudformation
- There is a plugin.

[serverless-plugin-custom-domain](#)

api.p9e.io
Uploaded on 8/3/2017

ACM Certificate (us-east-1 only)

api.p9e.io (1f96a962) ▾

Base Path Mappings

Path	Destination
experiment	production-bac... ▾ : production ▾
meter-data	production-met... ▾ : production ▾
standingdat	production-sta... ▾ : production ▾

[Add mapping](#)

Disk usage matters



- 50 MB if deploying directly.
- 250 if going from S3.
- We use Node.js, Webpack and *tree shaking* help us ([serverless webpack plugin](#))
- 75GB for entire region, covers all lambdas and versions of lambdas, you might need a [janitor lambda](#)...

Node.js Event loop

- We use postgres and connection pooling
- Event loop will never become empty
- Use Middy! :)

```
const middy = require('middy')
const {doNotWaitForEmptyEventLoop} = require('middy/middlewares')
const handler = middy((event, context, cb) => {
  // ...
}).use(doNotWaitForEmptyEventLoop())
```



S3 events: filename encoding



s3://podge-toys



```
{
  "Records": [{
    "s3": {
      "object": {
        "key": "Podge%27s+Unicorn.png"
      }
    }
  ]
}
```

Space replaced with "+" & URL encoded

```
const middy = require('middy')
const { s3KeyNormalizer } = require('middy/middlewares')
middy((event, context, cb) => {
  console.log(event.Records[0].s3.object.key) // Podge's Unicorn
}).use(s3KeyNormalizer())
```

In Summary...

- Not a great body of work out there on best practices (yet...)
- You trip over the simplest of things in AWS
- Local tooling still lacking
- We still believe **this to be the future** and it is definitely improving!





Thank you!



@Podgeypoos79



@loige

serverless  *lab*

Ad serverlesslab.com ▼