

Serverless и React **2**

(ловкость рук и никакого мошенничества)

Marina Miranovich

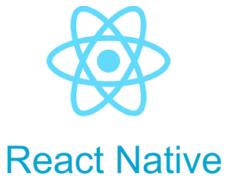
@obnoxious_mari

«ерат»

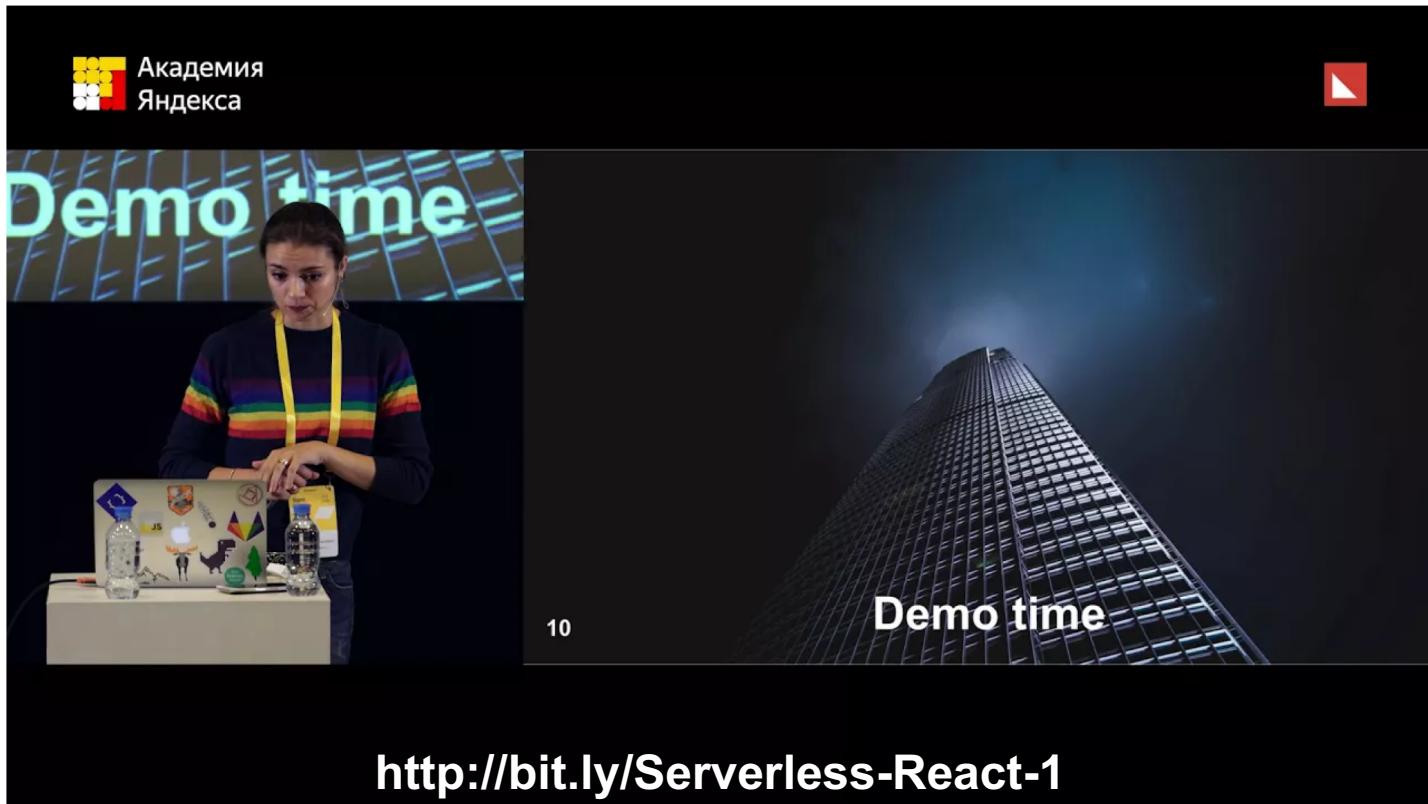


Nice to meet you!

Ведущий разработчик в <ерам>

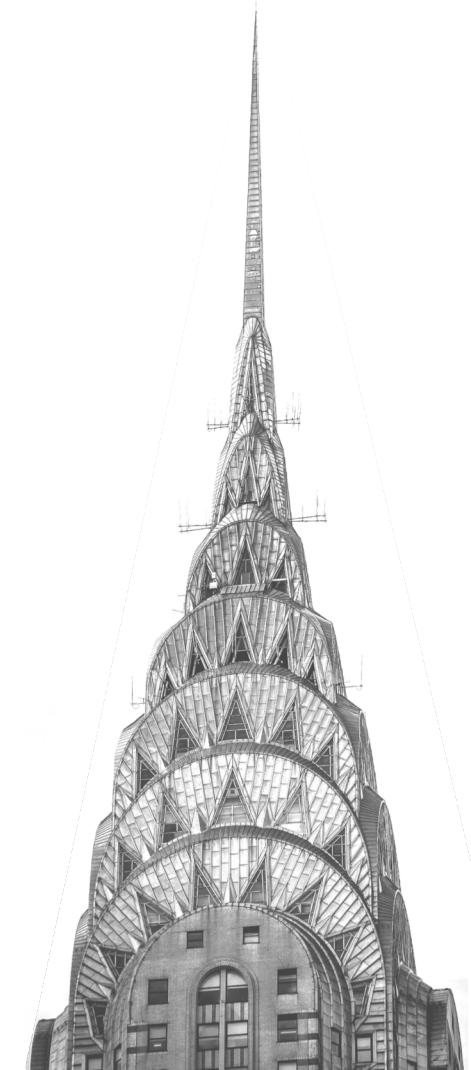


Первая серия - FrontTalks 2017



План!

1. Краткое описание первой части
2. Изоморфное React/Redux приложение на лямбде
3. Плюсы и минусы AWS лямбд
4. Для чего еще можно использовать лямбду?
5. Аналоги AWS Lambda



Что я буду сегодня использовать

S3 для хранения ассетов

IAM права доступа для пользователей и сервисов

API Gateway URL для доступа к сайту

CloudFormation для деплоя

AWS Lambda



Важно!



И это
бесплатно!

Что такое AWS Lambda?



Как деплоить лямбду

1. Не деплоить!



Как деплоить лямбду: Не деплоить!

The screenshot shows the AWS Lambda console interface. At the top, there are navigation links for Services, Resource Groups, and various account details. Below the header, the function name 'HolyJSDemo' is displayed, along with deployment and testing buttons. The configuration section includes dropdowns for Code entry type ('Edit code inline'), Runtime ('Node.js 6.10'), and Handler ('index.handler'). The main area is a code editor for 'index.js' containing the following code:

```
1 exports.handler = (event, context, callback) => {
2     // TODO implement
3     const response = `<html><head><title>Hello HolyJS</title></head><body><h1>Hello HolyJS!</h1>
4     console.log('Hi from the lambda!');
5     callback(null, {
6         statusCode: 200,
7         headers: {
8             "Content-Type": "text/html; charset=UTF-8"
9         },
10        body: response
11    });
12};
```

Как деплоить лямбду

1. Не деплоить!
2. Из командной строки загружать Zip



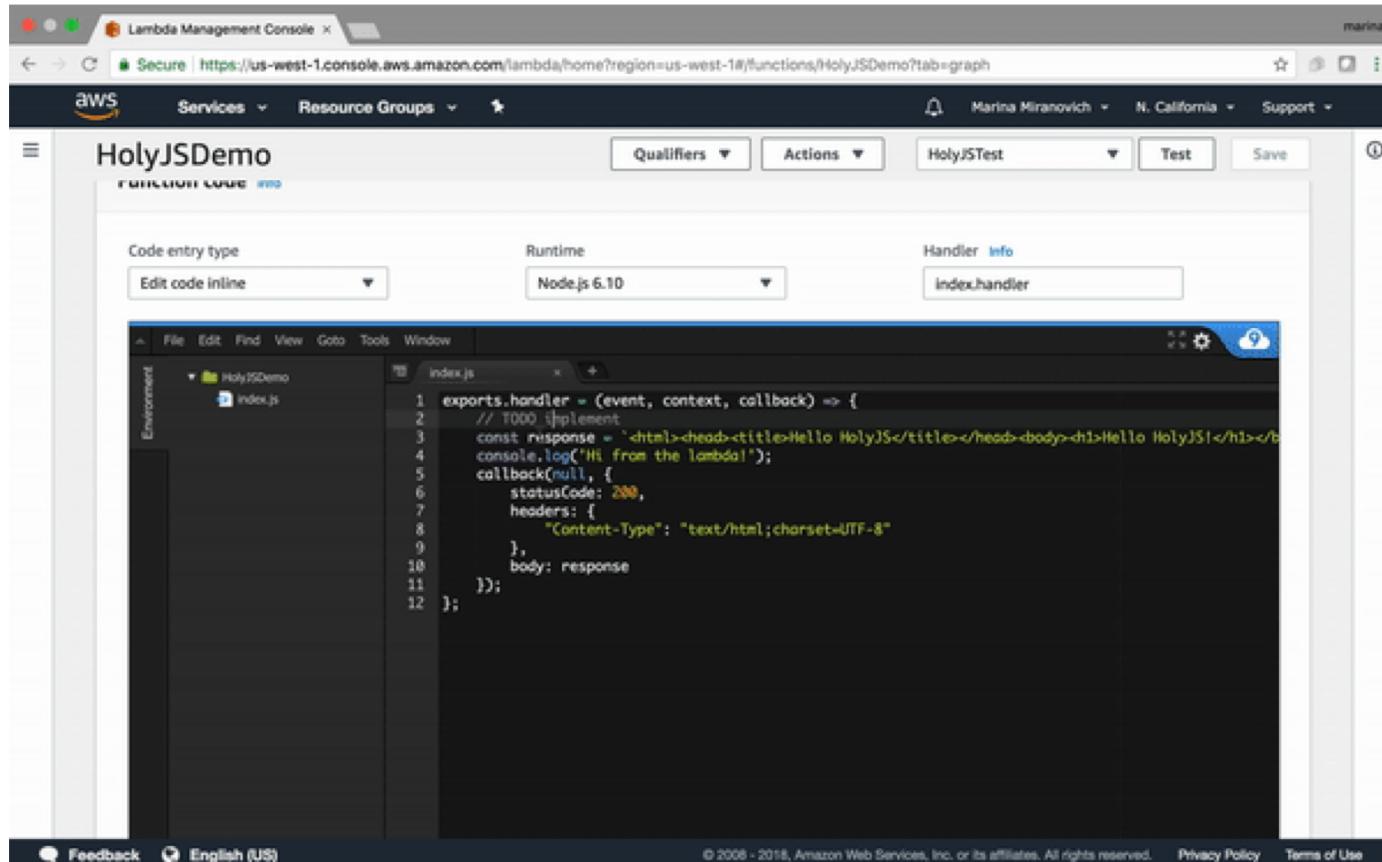
Как деплоить лямбду: Загружать Zip

```
zip -r build/lambda.zip index.js [node_modules/] [package.json]
```

```
aws lambda create-function ...
```

```
aws lambda update-function-code ...
```

Как деплоить лямбу: Загружать Zip



Как деплоить лямбду

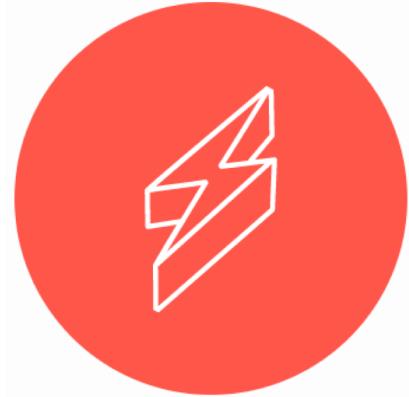
1. Не деплоить!
2. Из командной строки загружать Zip
3. Использовать фреймворк



Фреймворки для AWS Lambda

1. AWS CloudFormation & AWS CLI
2. APEX
3. Zappa (только для Python)
4. Claudia.js
5. Terraform 😕
6. Serverless 😊





Serverless

- Поддерживает сторонние плагины
- Локальная Lambda

The background features a minimalist architectural design with large, white, angular panels set against a clear blue sky. One panel is angled upwards and to the right, while another is curved downwards and to the left, creating a sense of dynamic movement. A smaller, grey rectangular structure is visible at the bottom left.

Demo Time

Запускаем лямбду локально

```
7
8 provider:
9
10
11
12 functions:
13   second:
14     handler: holyjs.hello
15     events:
16       - http:
17         method: get
18         path: holyjs
19
```

PROBLEMS TERMINAL ... 1: bash :

Marinas-MacBook-Pro:sls-holyjs marina\$

```
sls invoke local -f [fn_name]
```

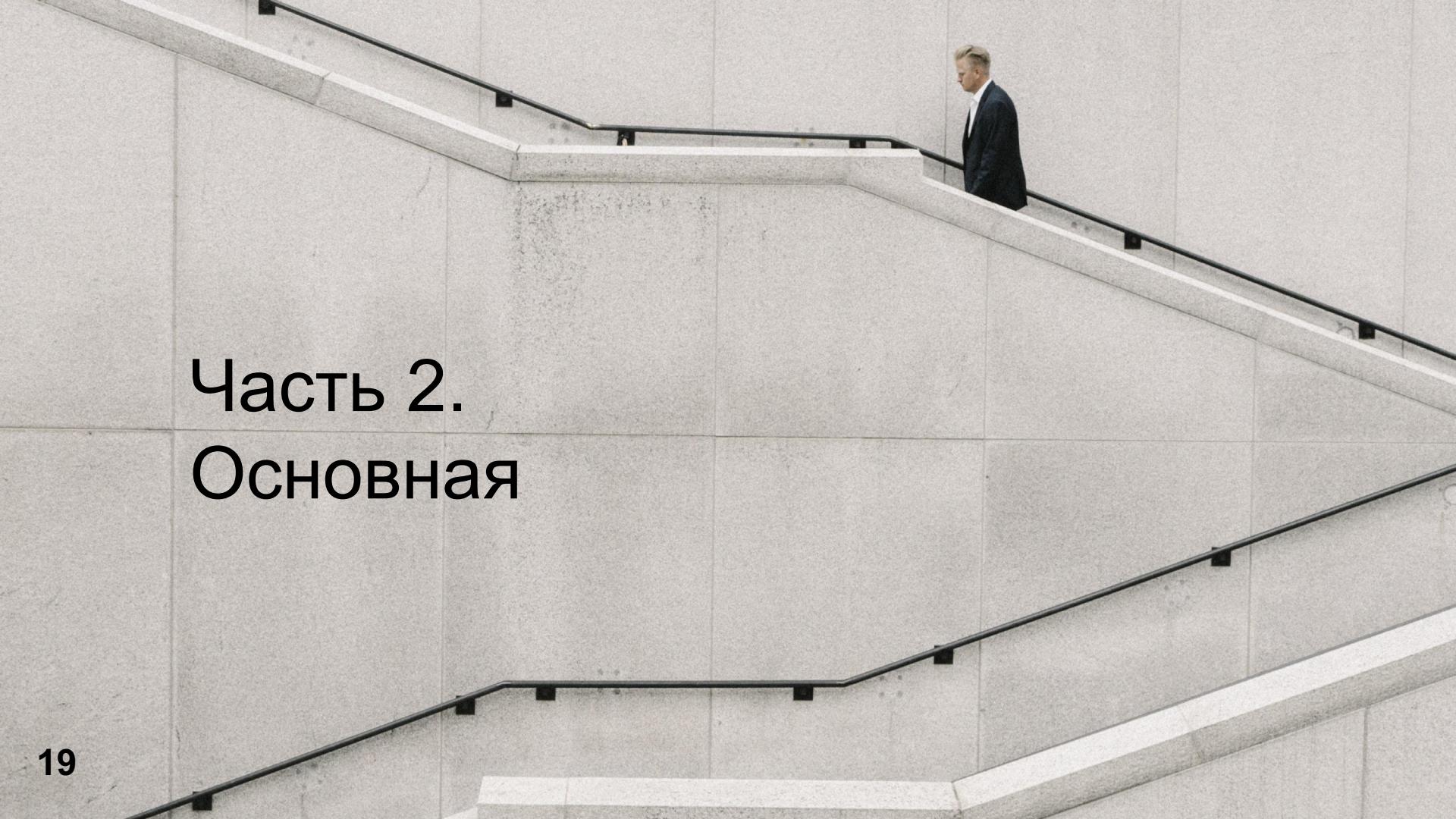
Запускаем лямбду локально serverless-offline



ПЛАГИНЫ

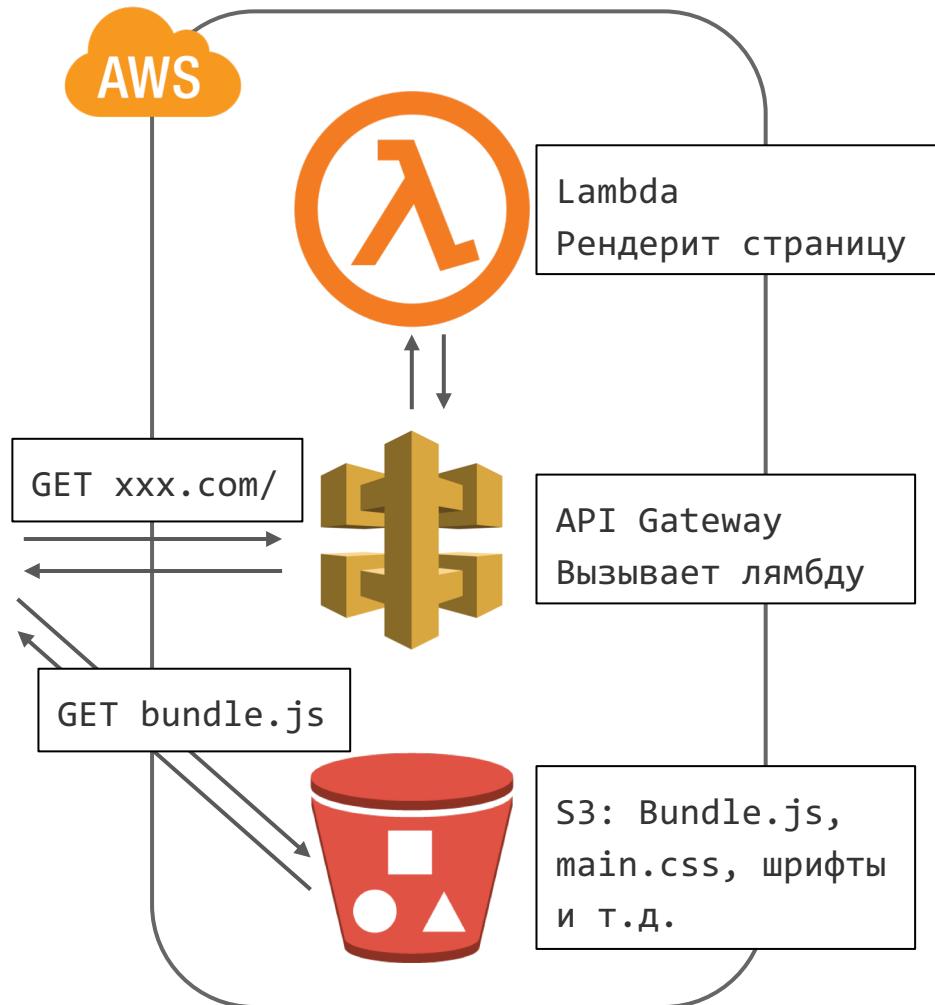
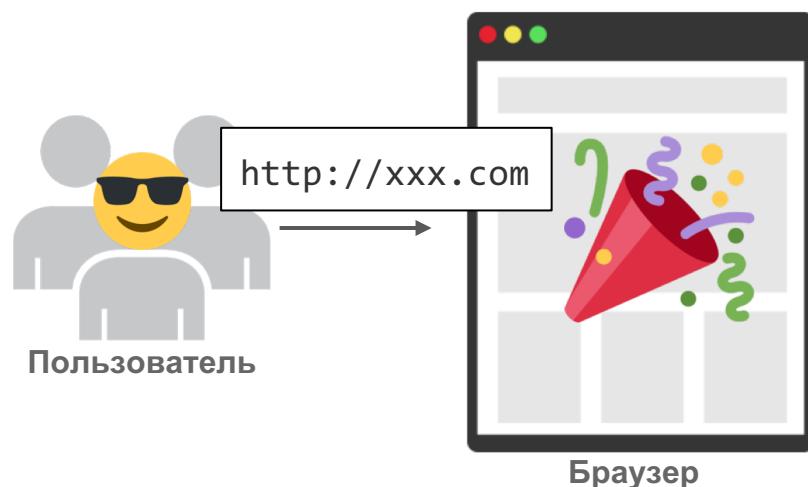


```
sls offline --port 8000 start
```

A man in a dark suit and white shirt is walking down a wide, light-colored stone staircase. The stairs are made of large, rectangular tiles, and black metal railings run along the sides. The background consists of large, light-colored stone walls.

Часть 2. Основная

Как это работает?



<https://github.com/Marina-Miranovich/lambda-react>

<http://bit.ly/react-lambda>

Структура проекта

lambda-react

```
├── README.md  
├── config  
├── package.json  
├── public  
├── scripts  
├── serverless.yml  
└── src  
    └── yarn.lock
```



Структура проекта

lambda-react

— README.md

```
config
├── package.json
├── public
├── scripts
├── serverless.yml
└── src
    └── yarn.lock
```

→

config

└── env.js

— jest

|- paths.js

└── polyfills.js

— webpack.config.dev.js

webpack.config.lambda.js

— webpack.config.prod.js

└ webpackDevServer.config.js



Lambda Webpack config

```
1 module.exports = {  
2   entry: slsw.lib.entries,  
3   target: 'node',  
4   module: {  
5     loaders: [  
6       { test: /\.js$/, loaders: ['babel-loader'], include: path.resolve(__dirname, '../src'), exclude: /node_modules/ },  
7       { test: /\.css$/, loaders: ['node-style-loader','css-loader?importLoaders=1'], exclude: /node_modules/ },  
8       { test: [/\png$/, /\jpg$/], loader: 'url-loader', options: { limit: 10000, mimetype: 'image/png' } }  
9     ],  
10   },  
11   output: {  
12     libraryTarget: 'commonjs',  
13     path: path.join(__dirname, '../webpack'),  
14     filename: '[name].js',  
15   },  
16 };
```

Структура проекта

lambda-react

```
├── README.md  
├── config  
├── package.json  
├── public  
├── scripts  
├── serverless.yml  
└── src  
    └── yarn.lock
```



package.json

```
1  {
2    "name": "lambda-react-demo",
3    "version": "1.0.0",
4    "scripts": {
5      "start": "node scripts/start.js",
6      "build": "node scripts/build.js",
7      "lambda": "sls offline --port 8000 start",
8      "deploy:lambda": "sls deploy"
9    },
10   "dependencies": {
...}
```

Структура проекта

lambda-react

```
├── README.md  
├── config  
├── package.json  
├── public  
├── scripts  
├── serverless.yml   
└── src  
    └── yarn.lock
```

serverless.yml

```
1 service:  
2   name: lambda-react-demo  
3 plugins:  
4   - serverless-webpack  
5   - serverless-offline  
6 provider:  
7   name: aws  
8   runtime: nodejs6.10  
9 functions:  
10  renderPage:  
11    handler: src/lambda/handler.renderPage  
12    events:  
13      - http:  
14          method: GET  
15          path: /  
16 custom:  
17    webpack: "./config/webpack.config.lambda.js"
```

Структура проекта

lambda-react

```
├── README.md  
├── config  
├── package.json  
├── public  
├── scripts  
├── serverless.yml  
└── src  
    └── yarn.lock
```



Структура проекта

lambda-react

- ─ README.md
- ─ config
- ─ package.json
- ─ public
- ─ scripts
- ─ serverless.yml

src

- ─ yarn.lock



src

- ─ actions
- ─ apis
- ─ components
- ─ index.css
- ─ index.js
- ─ **lambda**
- ─ reducers
- ─ sagas



Структура проекта

lambda-react

```
├── README.md  
├── config  
├── package.json  
├── public  
├── scripts  
├── serverless.yml  
└── src  
    ├── actions  
    ├── apis  
    ├── components  
    ├── index.css  
    └── index.js  
  
└── lambda  
    ├── reducers  
    └── sagas  
yarn.lock
```



lambda

```
└── handler.js  
    └── renderer.js
```



Lambda handler src/lambda/handler.js

```
1 import render from './renderer';
2
3 const createResponse = (html) => {
4   return {
5     statusCode: 200,
6     headers: {
7       "Content-Type": "text/html; charset=utf-8"
8     },
9     body: html
10  };
11};
12
13 export const renderPage = (event, context, cb) => {
14   render()
15     .then((html) => cb(null, createResponse(html)))
16     .catch(e => cb(e));
```

Lambda handler src/lambda/handler.js

```
1 import render from './renderer';
2
3 const createResponse = (html) => {
4     return {
5         statusCode: 200,
6         headers: {
7             "Content-Type": "text/html; charset=utf-8"
8         },
9         body: html
10    };
11 };
12
13 export const renderPage = (event, context, cb) => {
14     render()
15         .then((html) => cb(null, createResponse(html)))
16         .catch(e => cb(e));
17 };
```

Lambda handler src/lambda/handler.js

```
1 import render from './renderer';
2
3 const createResponse = (html) => {
4   return {
5     statusCode: 200,
6     headers: {
7       "Content-Type": "text/html; charset=utf-8"
8     },
9     body: html
10  };
11};
12
13 export const renderPage = (event, context, cb) => {
14   render()
15     .then((html) => cb(null, createResponse(html)))
16     .catch(e => cb(e));
```

Структура проекта

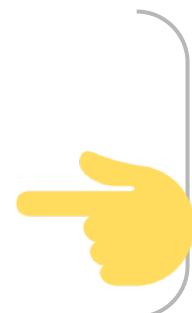
lambda-react

```
├── README.md  
├── config  
├── package.json  
├── public  
├── scripts  
├── serverless.yml  
└── src  
    ├── actions  
    ├── apis  
    ├── components  
    ├── index.css  
    └── index.js  
  
└── lambda  
    ├── reducers  
    └── sagas  
yarn.lock
```



lambda

```
└── handler.js  
    └── renderer.js
```



Что происходит в src/lambda/renderer.js

1. Изоморфное приложение или рендерим на сервере (на любом):

a. Рендерим React и Redux в HTML строку



ReactCasts #13 - Server Side
Rendering: Data Fetching &



ReactCasts #12 - Server Side
Rendering



Server Rendering with React
and React Router v4

Рендерим приложение src/lambda/renderer.js

```
1  const markup = renderToString(  
2      <Provider store={store}>  
3          <App />  
4      </Provider>);
```

Что происходит в src/lambda/renderer.js

1. Изоморфное приложение или рендерим на сервере (на любом):

- a. Рендерим React и Redux в HTML строку
- b. Добавляем стили



Стили renderer.js

```
1 const styleTag = collectInitial(); // from 'node-style-loader'  
2 const generateHTML = (markup, styleTag) => (  
3   `<!doctype html>  
4   <html lang="en">  
5     <head>  
6       ${styleTag}  
7     </head>
```

Что происходит в src/lambda/renderer.js

1. Изоморфное приложение или рендерим на сервере (на любом):

- a. Рендерим React и Redux в HTML строку
- b. Добавляем стили
- c. Передаем Redux state

Передаем данные src/lambda/renderer.js

```
1  const initialState = {...};  
...  
2  <body>  
3    <script>  
4      window.__PRELOADED_STATE__ = ${  
5        JSON.stringify(initialState).replace(/</g, '<\u003c')}  
6    </script>  
7  </body>
```

Что происходит в src/lambda/renderer.js

1. Изоморфное приложение или рендерим на сервере (на любом):

- a. Рендерим React и Redux в HTML строку
- b. Добавляем стили
- c. Передаем Redux state

2. Добавляем обработку ошибок (must have для лямбды)

Рендерим приложение src/lambda/renderer.js

```
1 const render = () => {  
2     return new Promise((resolve) => {
```

Что происходит в src/lambda/renderer.js

1. Изоморфное приложение или рендерим на сервере (на любом):

- a. Рендерим React и Redux в HTML строку
- b. Добавляем стили
- c. Передаем Redux state

2. Добавляем обработку ошибок (must have для лямбды)

3. Определяем где находимся

JS Bundle src/lambda/renderer.js

```
<script type="text/javascript" src="${bundleUrl}"></script>
...
const bundleUrl = process.env.NODE_ENV === 'AWS' ?
    AWS_URL : LOCAL_URL;
```

JS Bundle serverless.yml

```
...
1 functions:
2   renderPage:
3     handler: src/lambda/handler.renderPage
4     environment:
5       NODE_ENV: "AWS"
...
...
```

Что происходит в src/lambda/renderer.js

1. Изоморфное приложение или рендерим на сервере (на любом):

- a. Рендерим React и Redux в HTML строку
- b. Добавляем стили
- c. Передаем Redux state

2. Добавляем обработку ошибок (must have для лямбды)

3. Определяем где находимся

Приятный бонус:

Деплоим ассеты в S3 bucket.

Static assets serverless.yml



ПЛАГИНЫ

```
plugins:  
  - serverless-finch  
...  
custom:  
  client:  
    bucketName: lambda-react-demo
```



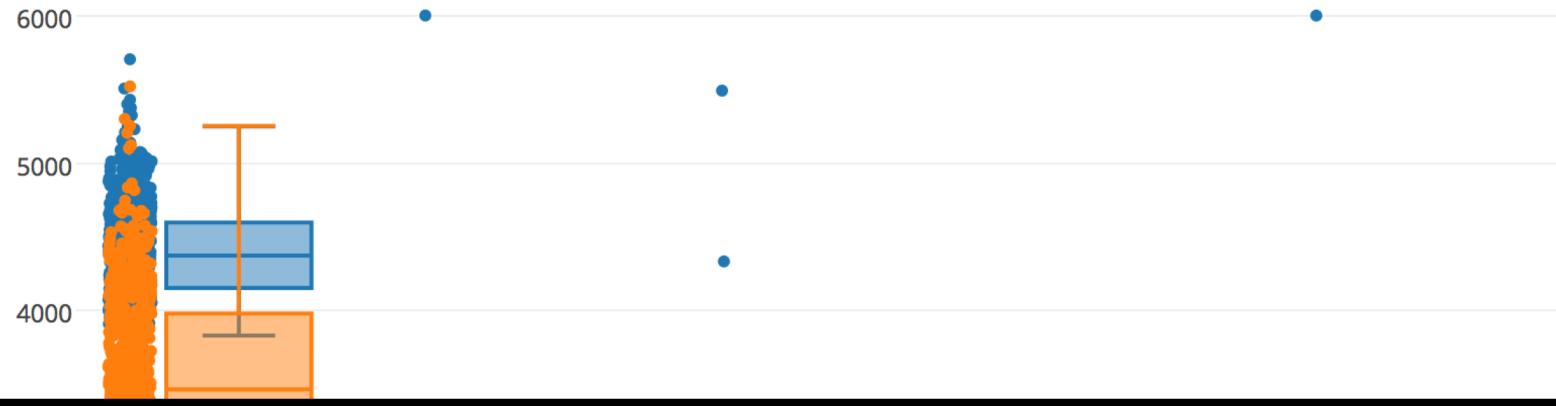


Минусы лямбда функций

Не время холодного старта! (для Node.js)



csharp
java
python
nodejs6



<https://read.acloud.guru/does-coding-language-memory-or-package-size-affect-cold-starts-of-aws-lambda-a15e26d12c76>



Минусы лямба функций

- Ограничения по размеру кода функции (**ZIP <= 50Mb**)

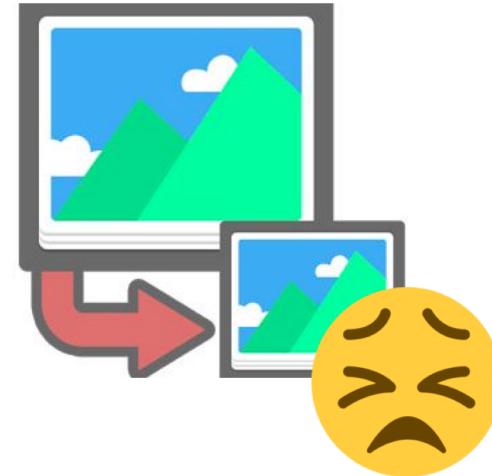
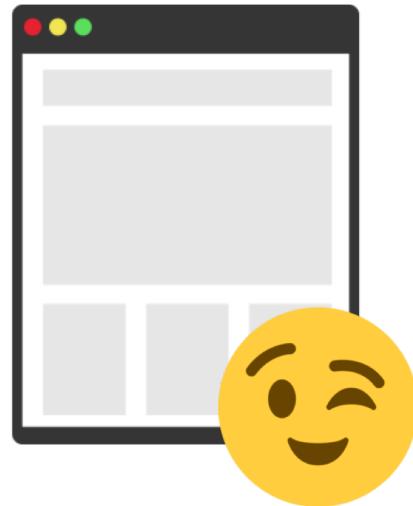


node_modules

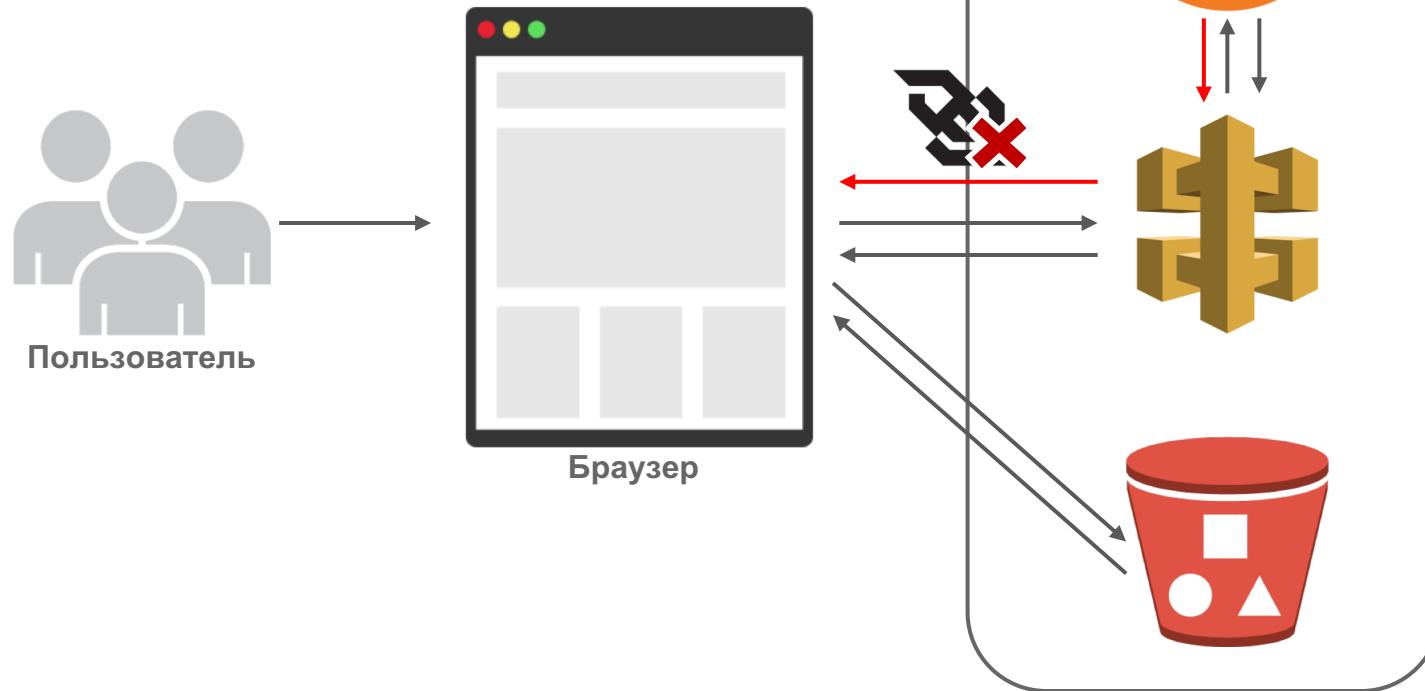


Минусы лямба функций

- Ограничения по размеру кода функции
- Ограничения по времени выполнения ($\leq 5\text{мин}$)



WebSockets?

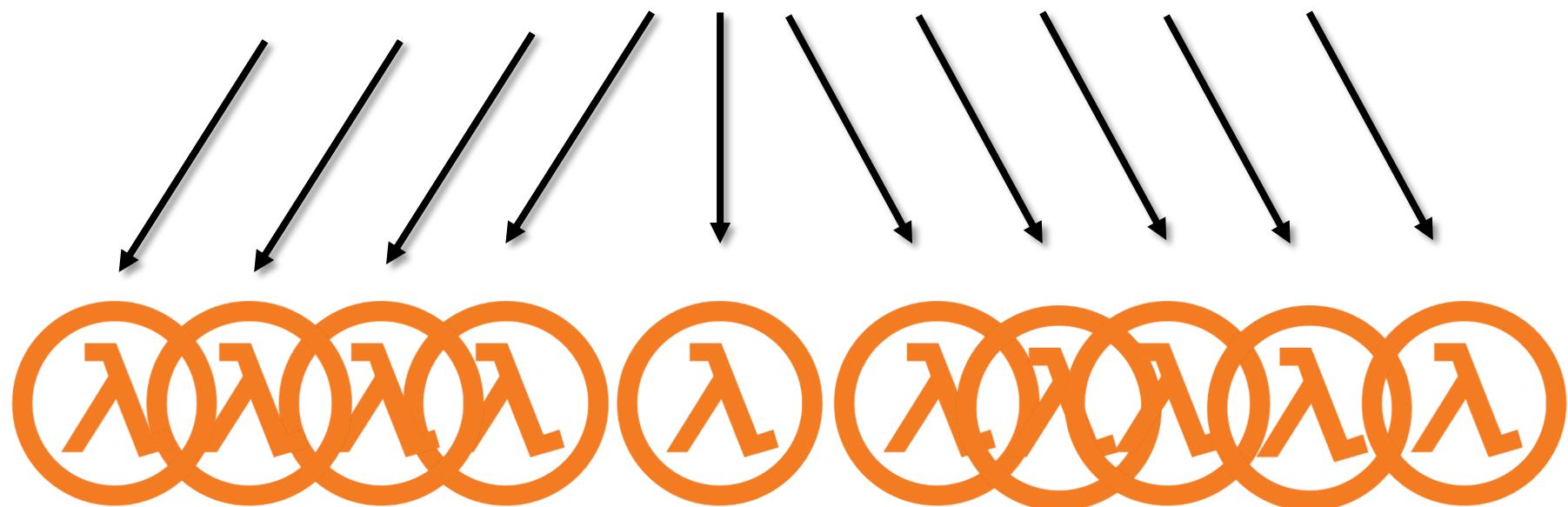


Минусы лямба функций

- Ограничения по размеру кода функции
- Ограничения по времени выполнения
- Количество параллельных функций - **500 - 3000**

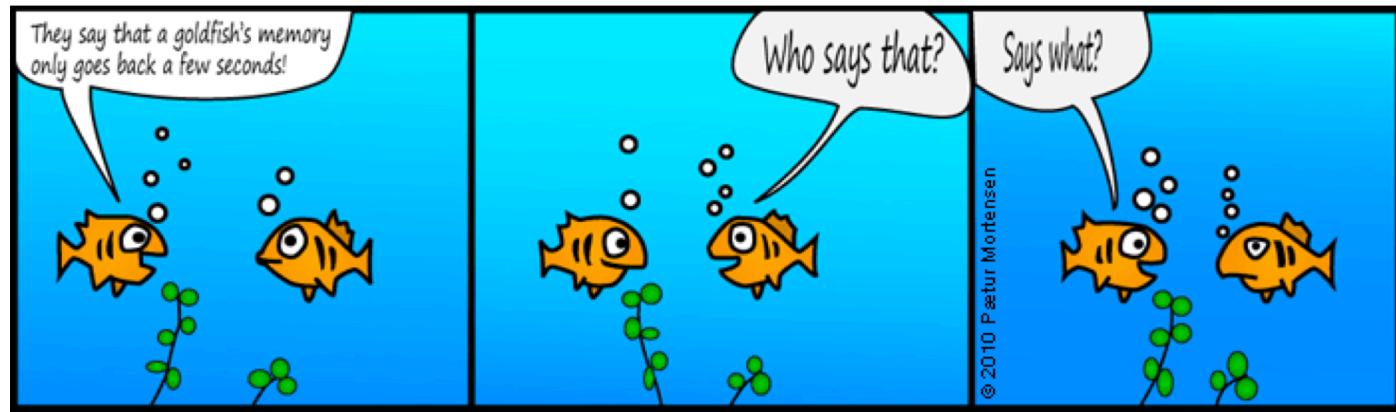


Маштабирование лямбды



Минусы лямба функций

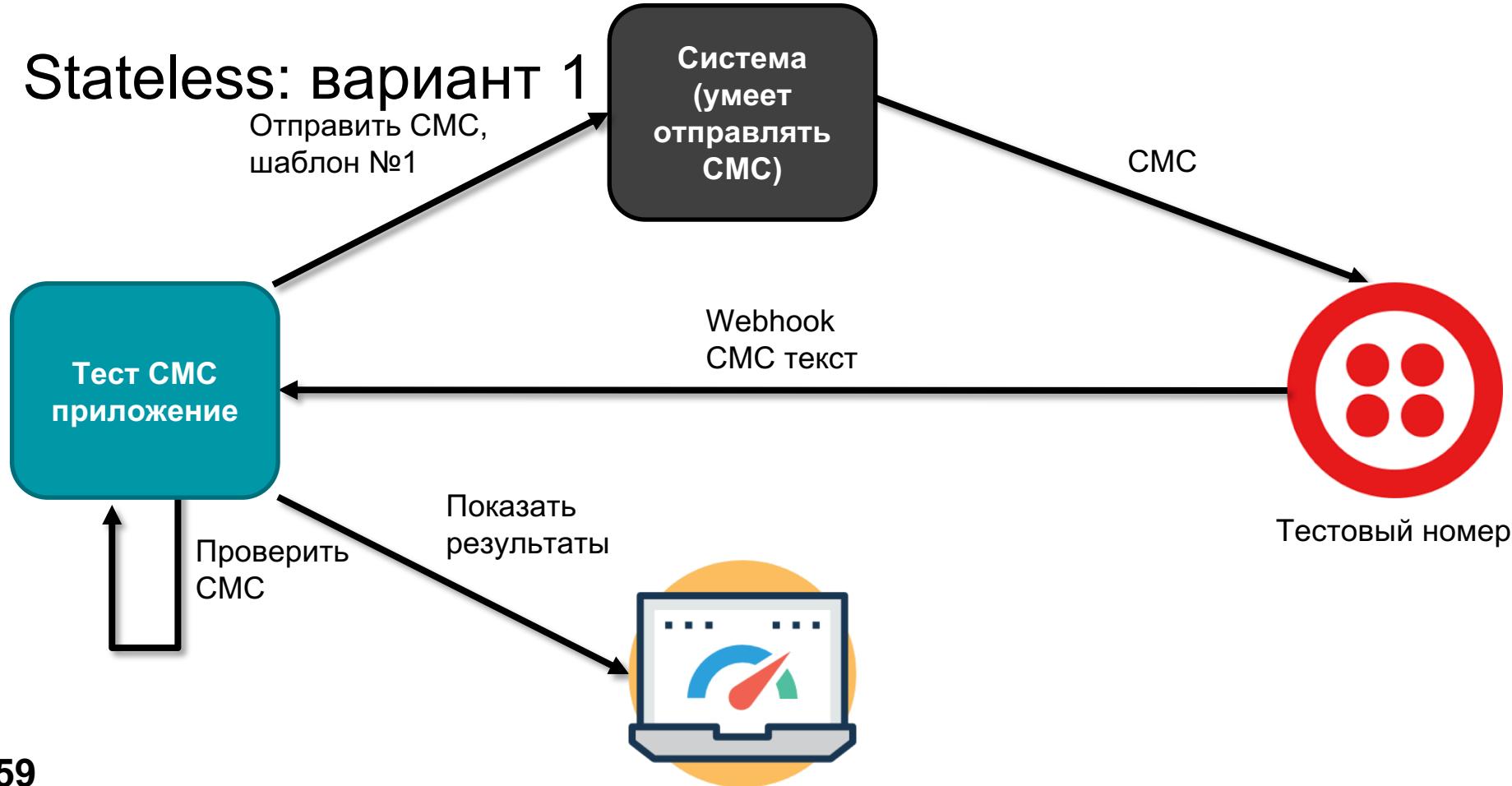
- Ограничения по размеру кода функции
- Ограничения по времени выполнения
- Количество параллельных функций
- Stateless



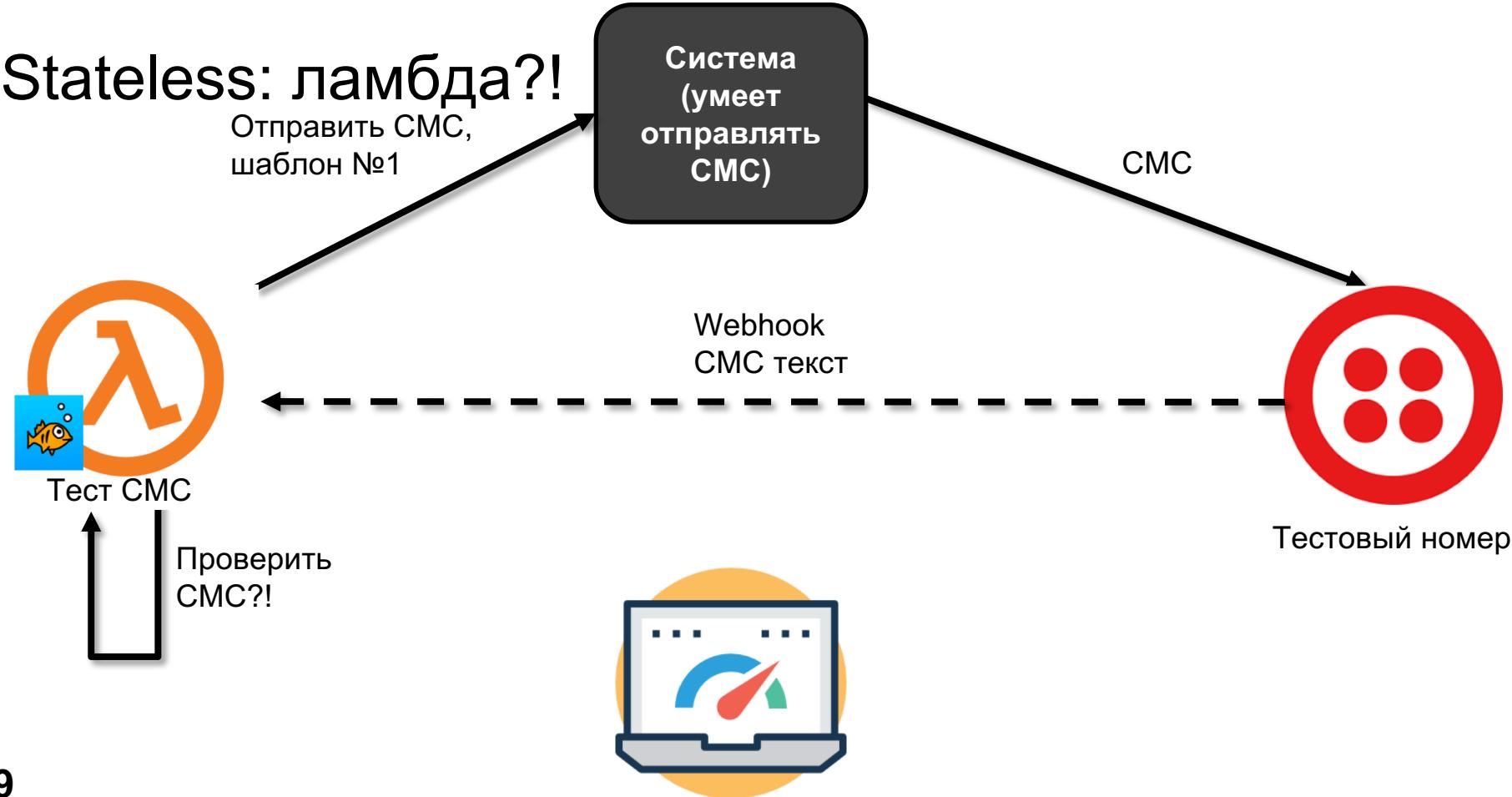
Stateless: Протестировать отправку СМС



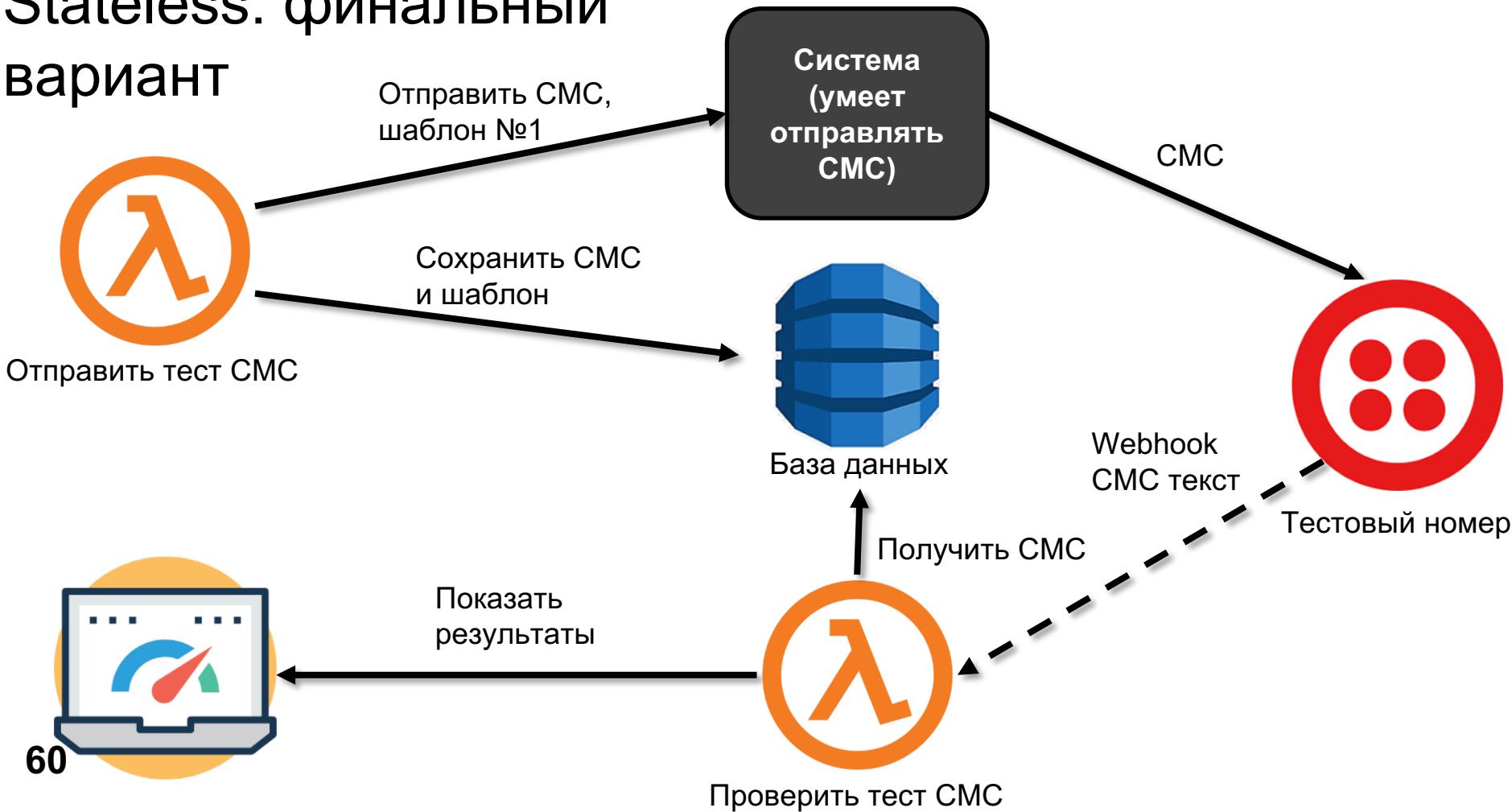
Stateless: вариант 1



Stateless: ламбда?!



Stateless: финальный вариант



Плюсы лямбда функций

- Действительно можно забыть о том, что есть сервера;
- Не надо думать о масштабировании;
- Легко интегрируется со многими сервисами на AWS (DynamoDB, Alexa, API Gateway, и т. д.);



Что еще можно
сделать на лямбдах?

Все

что

угодно*



Что еще можно сделать на лямбдах?

HTTP
services

IoT

Chat Bots

Image/Video
conversions

Machine
Learning

Batch Jobs



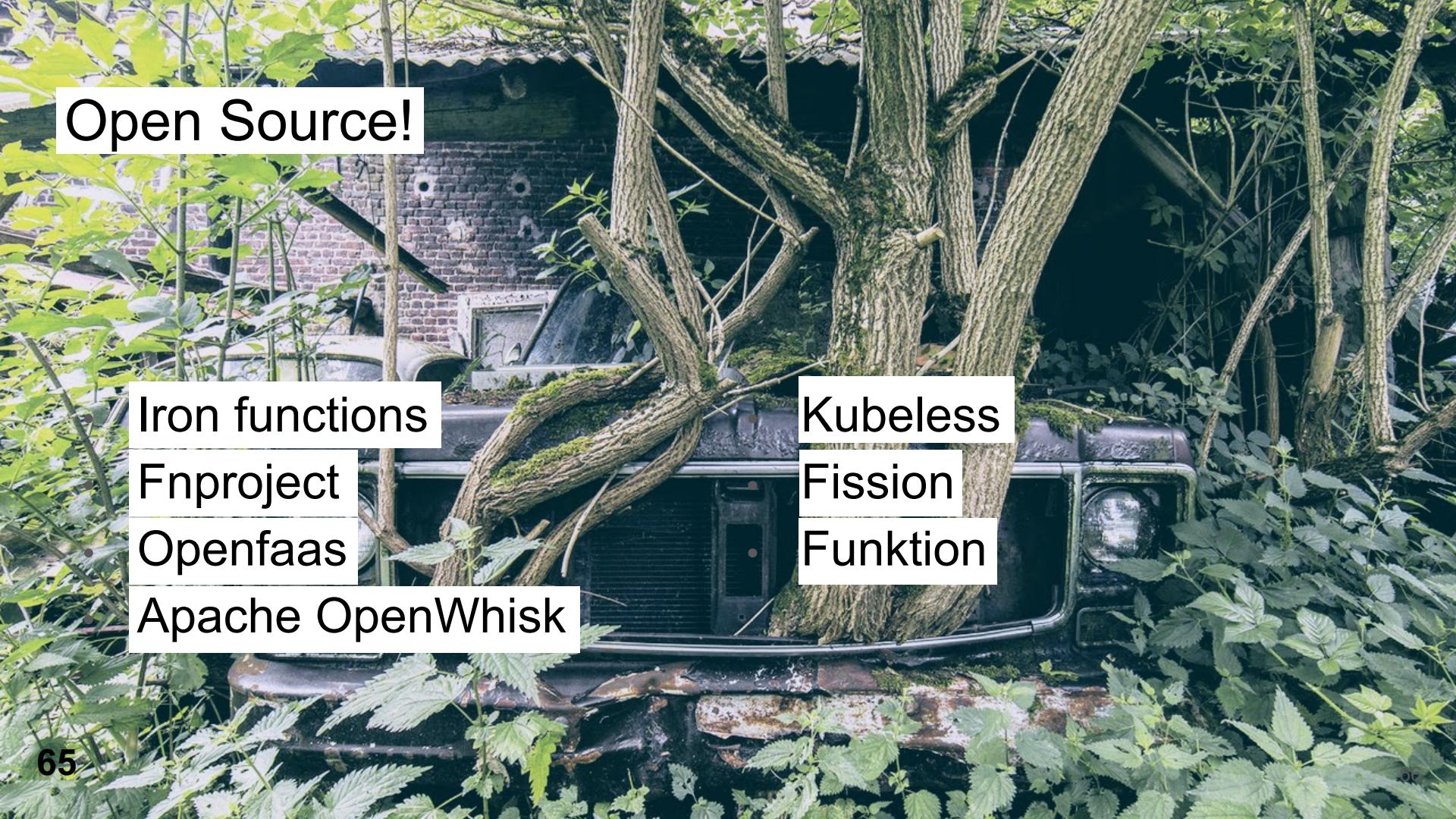
Есть ли жизнь после амазон?

Google

Azure

Twillio

IBM

A photograph of a dark-colored car, possibly black or dark blue, that has been abandoned and left to decay. It is heavily covered in thick, green ivy and other climbing plants, particularly on the front half. The car is positioned in a field of similar green plants, creating a sense of being overtaken by nature.

Open Source!

Iron functions

Fnproject

Openfaas

Apache OpenWhisk

Kubeless

Fission

Funktion

Спасибо!



Marina-Miranovich



obnoxious_marie



@obnoxious_mari