Serverless и React 2

(ловкость рук и никакого мошенничества)

Marina Miranovich
@obnoxious_mari
<epam>

Photo by Alex Block

Nice to meet you!

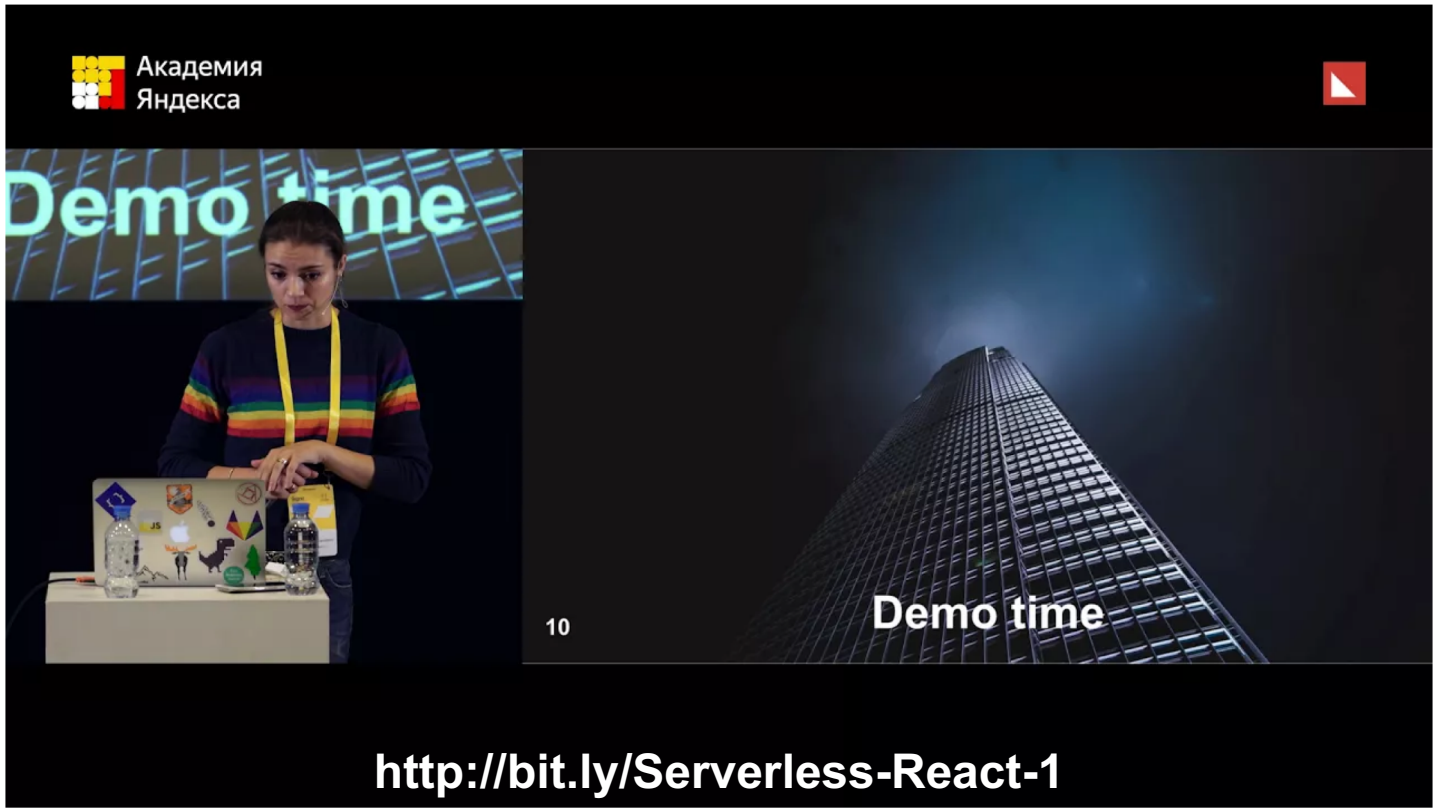
Ведущий разработчик в <epam>

AWS

React Native

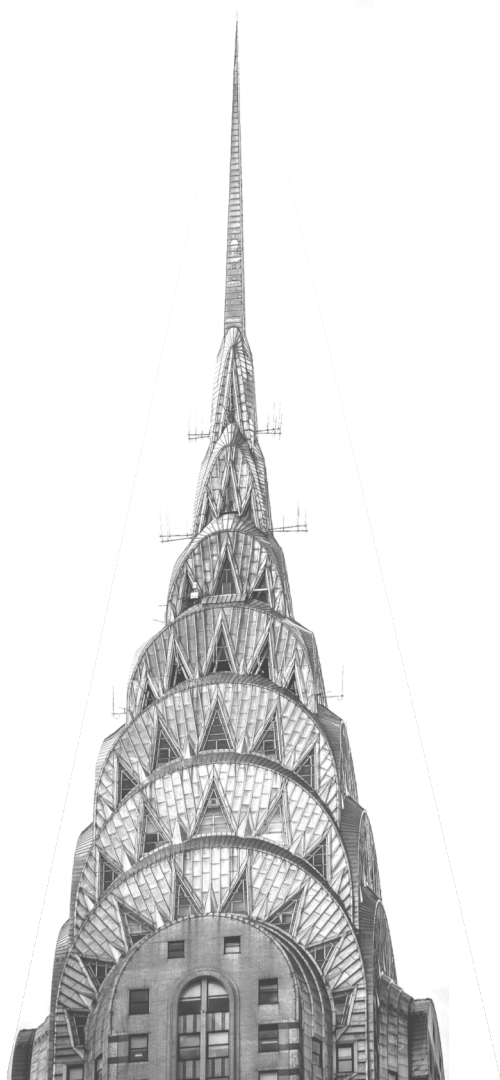
UNIFIED MODELING LANGUAGE™ UML®

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



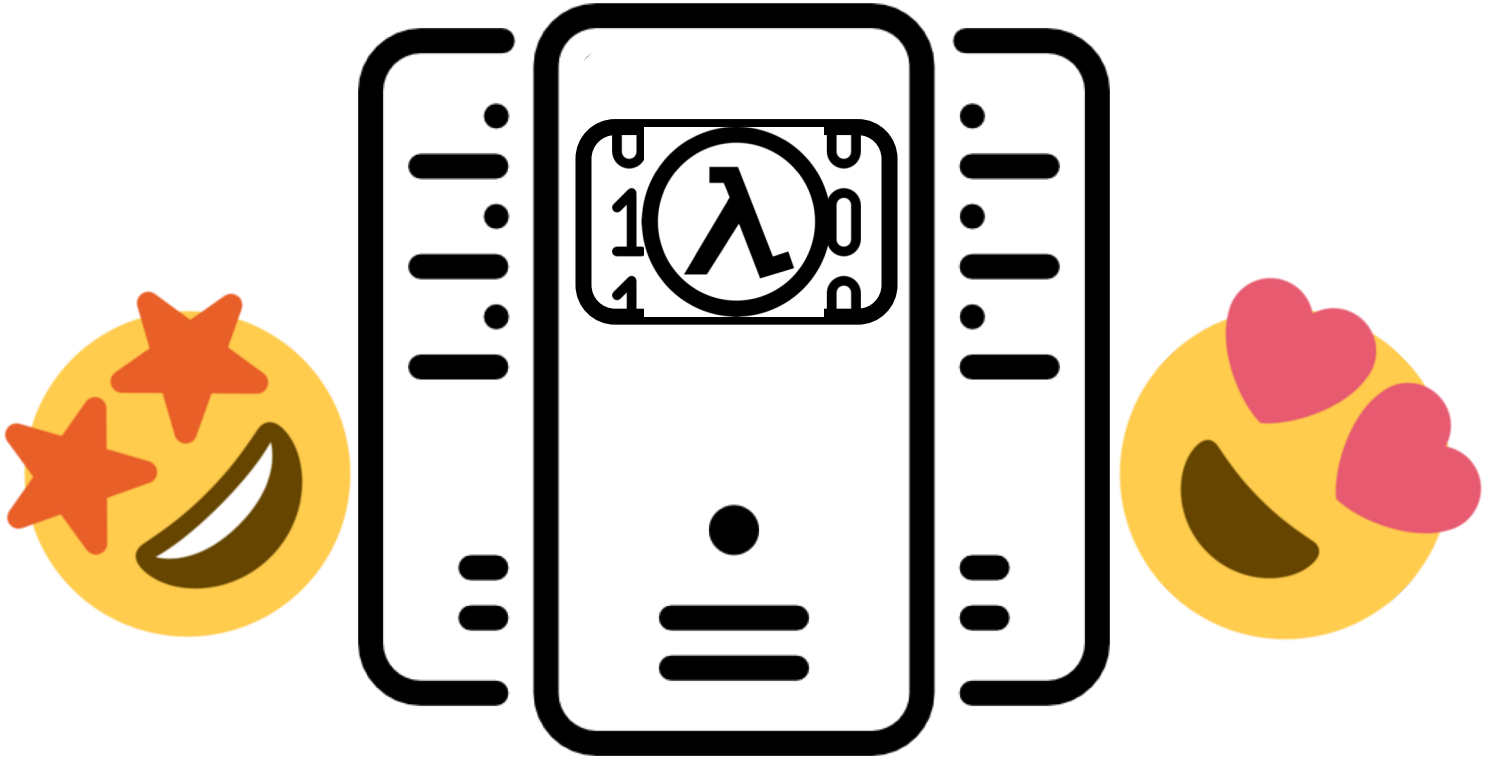
Что я буду сегодня использовать

S3	для хранения ассетов
IAM	права доступа для пользователей и сервисов
API Gateway	URL для доступа к сайту
CloudFormation	для деплоя
AWS Lambda	

Важно!

Все демо рекомендуются к самостоятельному повторению.*

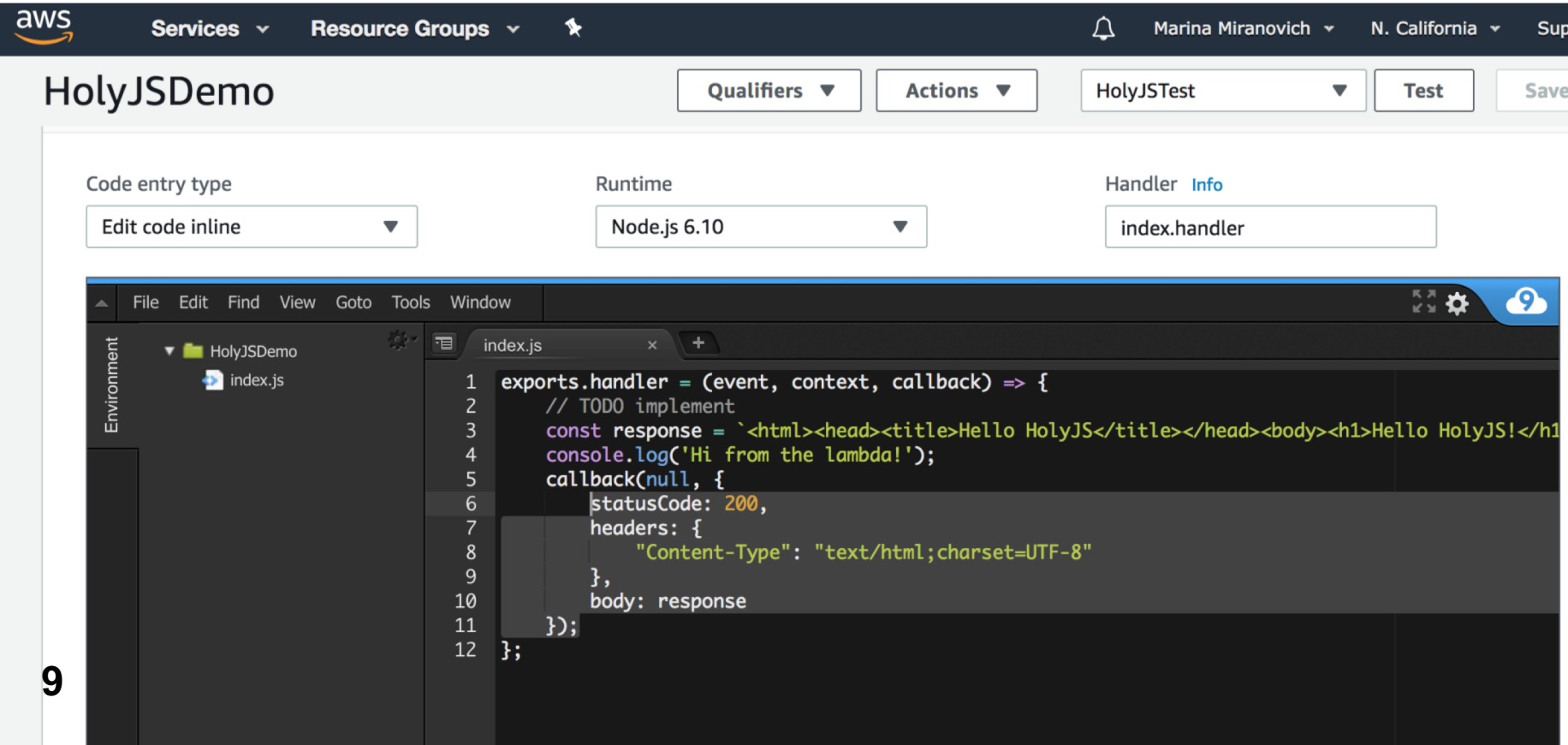
Что такое AWS Lambda?



Как деплоить лямбду

1. Не деплоить!

Как деплоить лямбду: Не деплоить!



Как деплоить лямбду

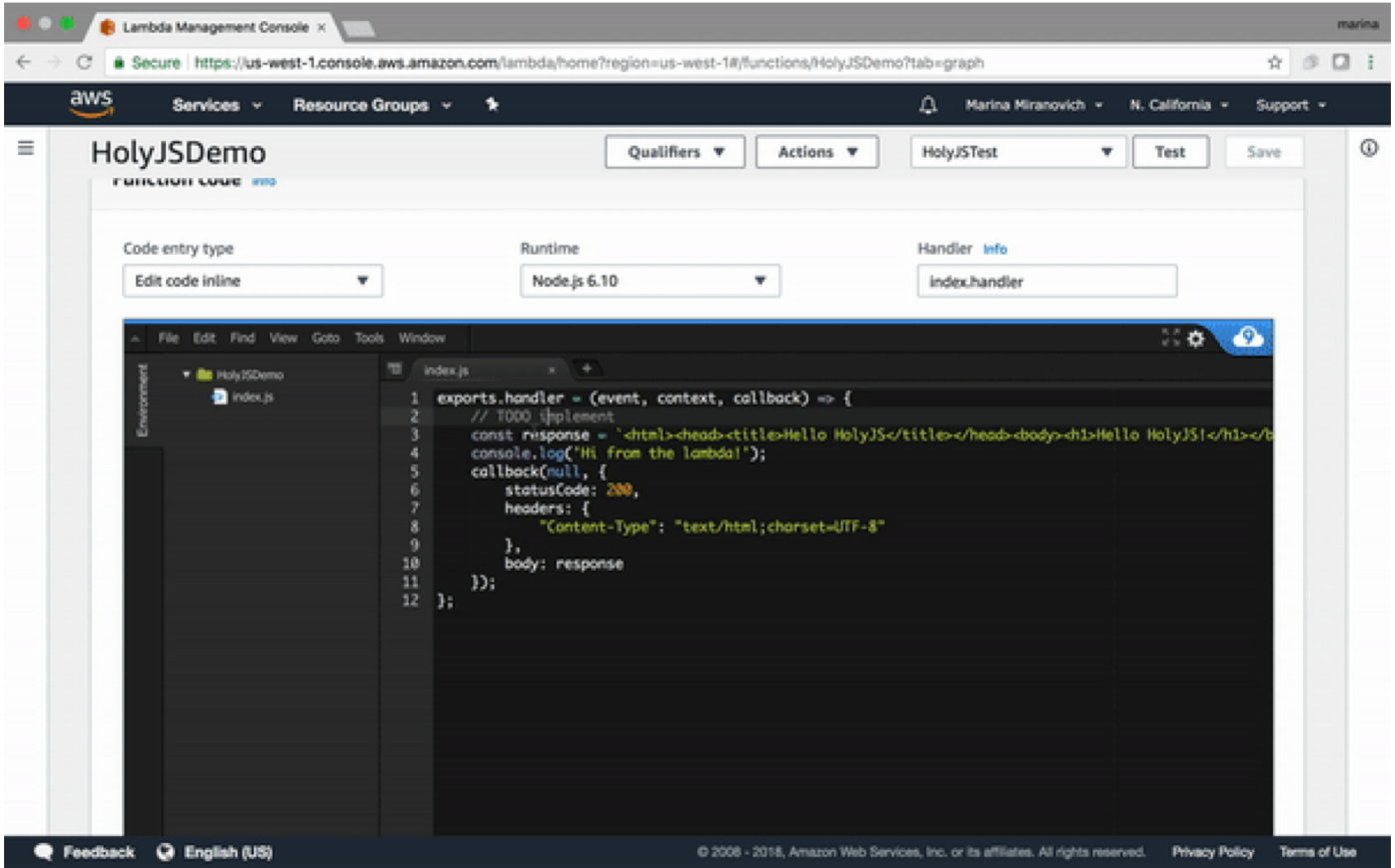
1. Не деплоить!
2. Из командной строки загружать Zip

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 лямбду: Загружать Zip



Как деплоить лямбду

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

Фреймворки для AWS Lambda

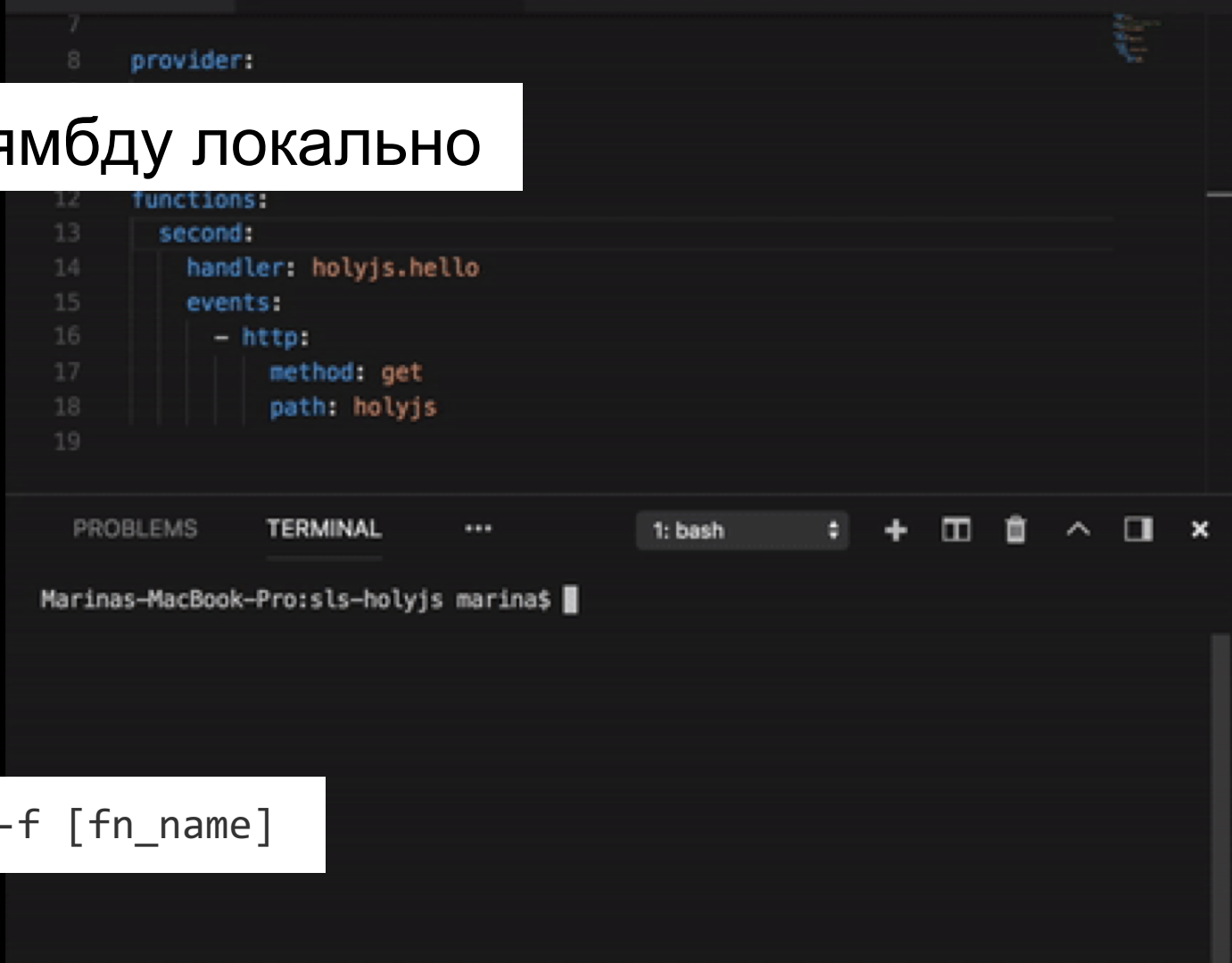
1. AWS CloudFormation & AWS CLI
2. APEX
3. Zappa (только для Python)
4. Claudia.js
5. Terraform 😳
6. Serverless 😃

Serverless

- Поддерживает сторонние плагины
- Локальная Lambda


Demo Time

Запускаем лямбду локально

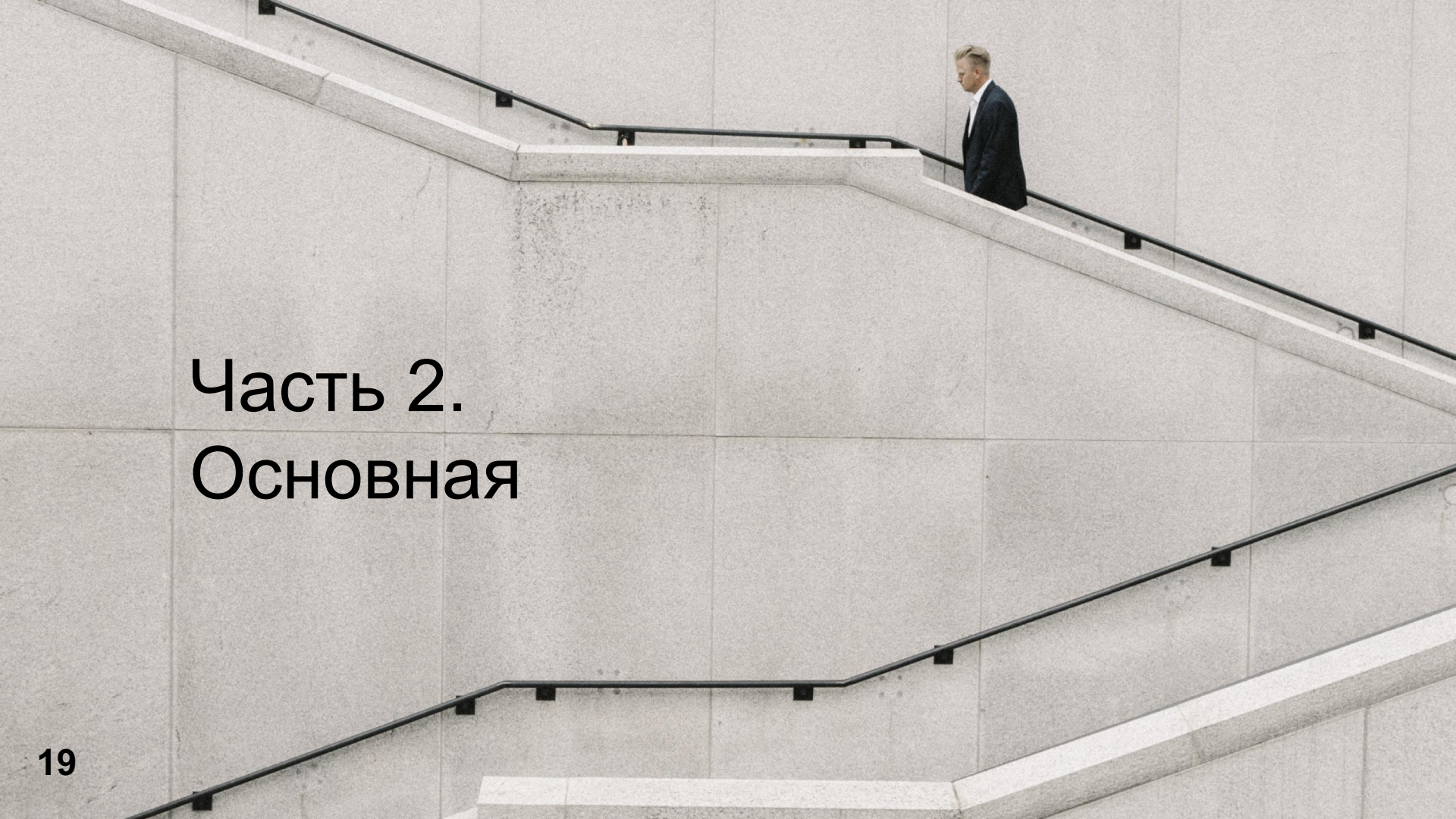



```
sls invoke local -f [fn_name]
```

Запускаем лямбду локально `serverless-offline`

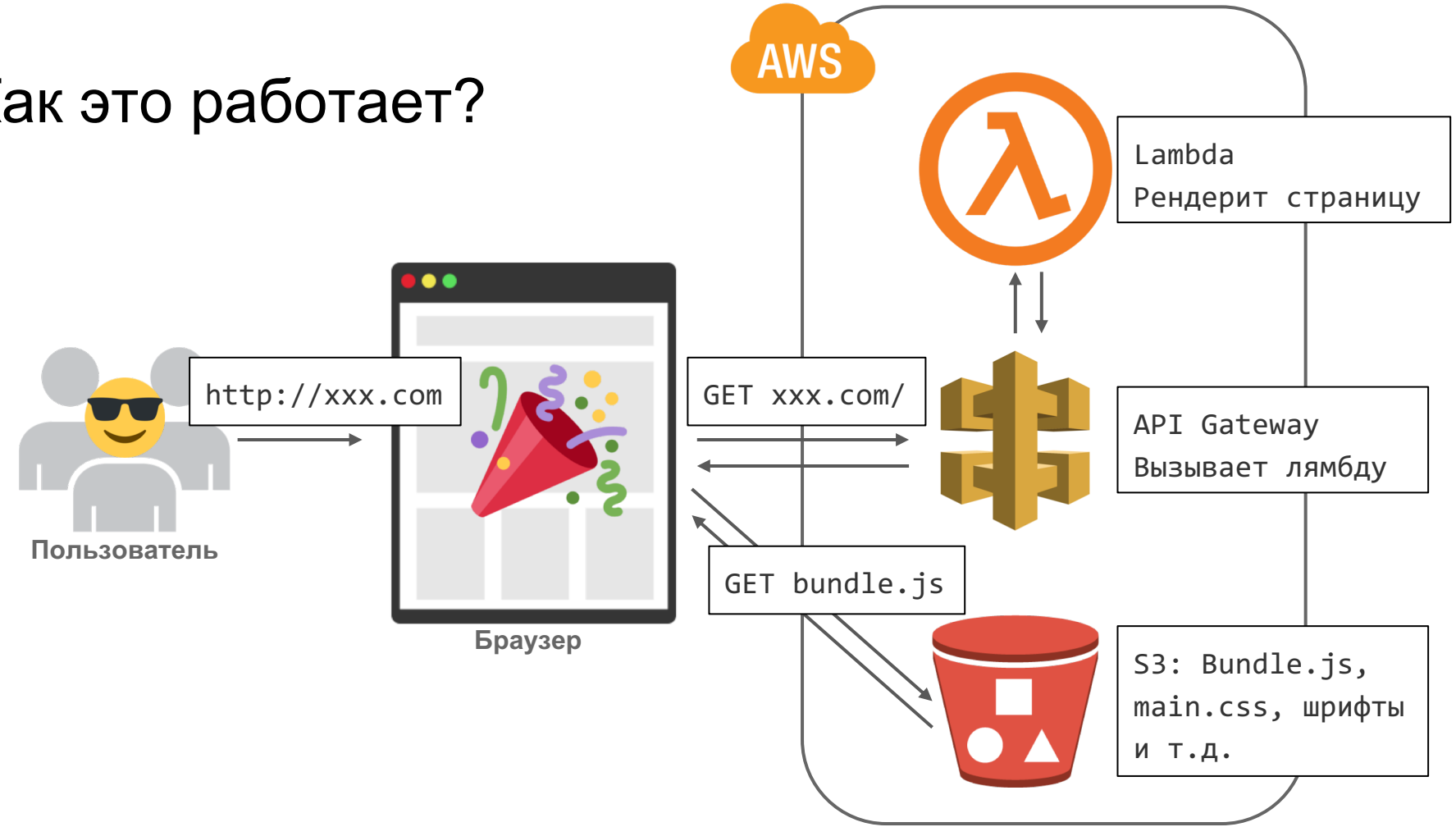
ПЛАГИНЫ

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



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

```
module.exports = {
  entry: slsw.lib.entries,
  target: 'node',
  module: {
   loaders: [
      { test: /\.js$/, loaders: ['babel-loader'], include: path.resolve(__dirname, '../src'), exclude: /node_modules/ },
      { test: /\.css$/, loaders: ['node-style-loader','css-loader?importLoaders=1'], exclude: /node_modules/ },
      { test: [/\.png$/, /\.jpg$/], loader: 'url-loader', options: { limit: 10000, mimetype: 'image/png' } }
    ],
  },
 output: {
   libraryTarget: 'commonjs',
   path: path.join(__dirname, '../.webpack'),
   filename: '[name].js',
  },
};
```

Структура проекта

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

package.json

```
{
 "name": "lambda-react-demo",
 "version": "1.0.0",
  "scripts": {
    "start": "node scripts/start.js",
    "build": "node scripts/build.js",
    "lambda": "sls offline --port 8000 start",
    "deploy:lambda": "sls deploy"
  },
 "dependencies": {
...
```

Структура проекта

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

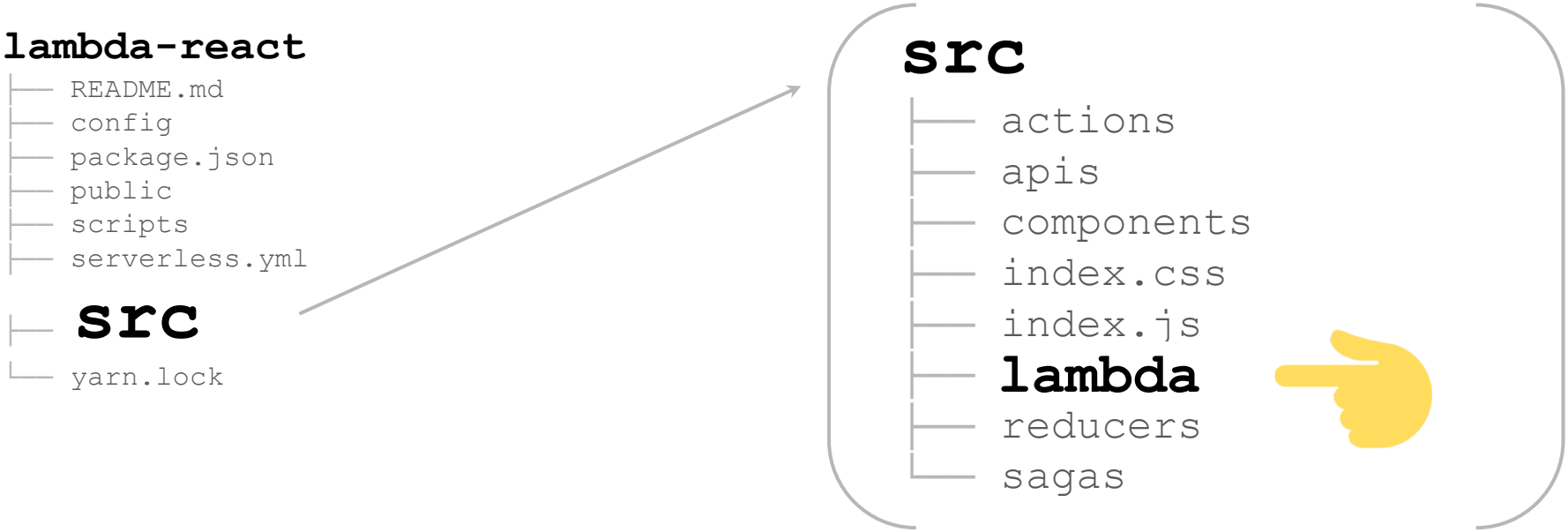
serverless.yml

```
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



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`

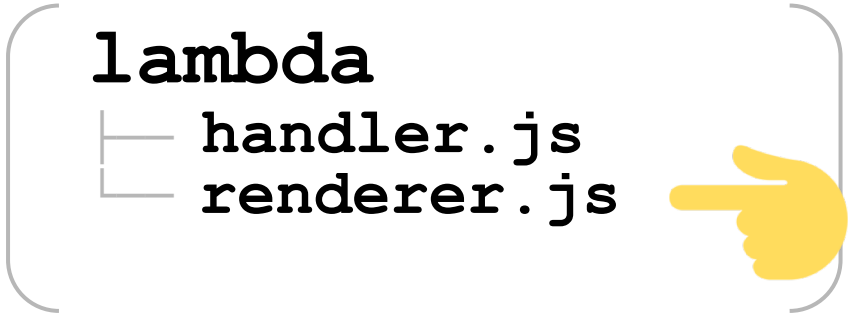
```
1  import render from './renderer';
2
3  const createResponse = (html) => {
4     return {
5        statusCode: 200,
6        headers: {
7           "Content-Type": "text/html; charset=utf-8"
8        },
9        body: html
10  };
11 };
12
13 export const renderPage = (event, context, cb) => {
14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



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`

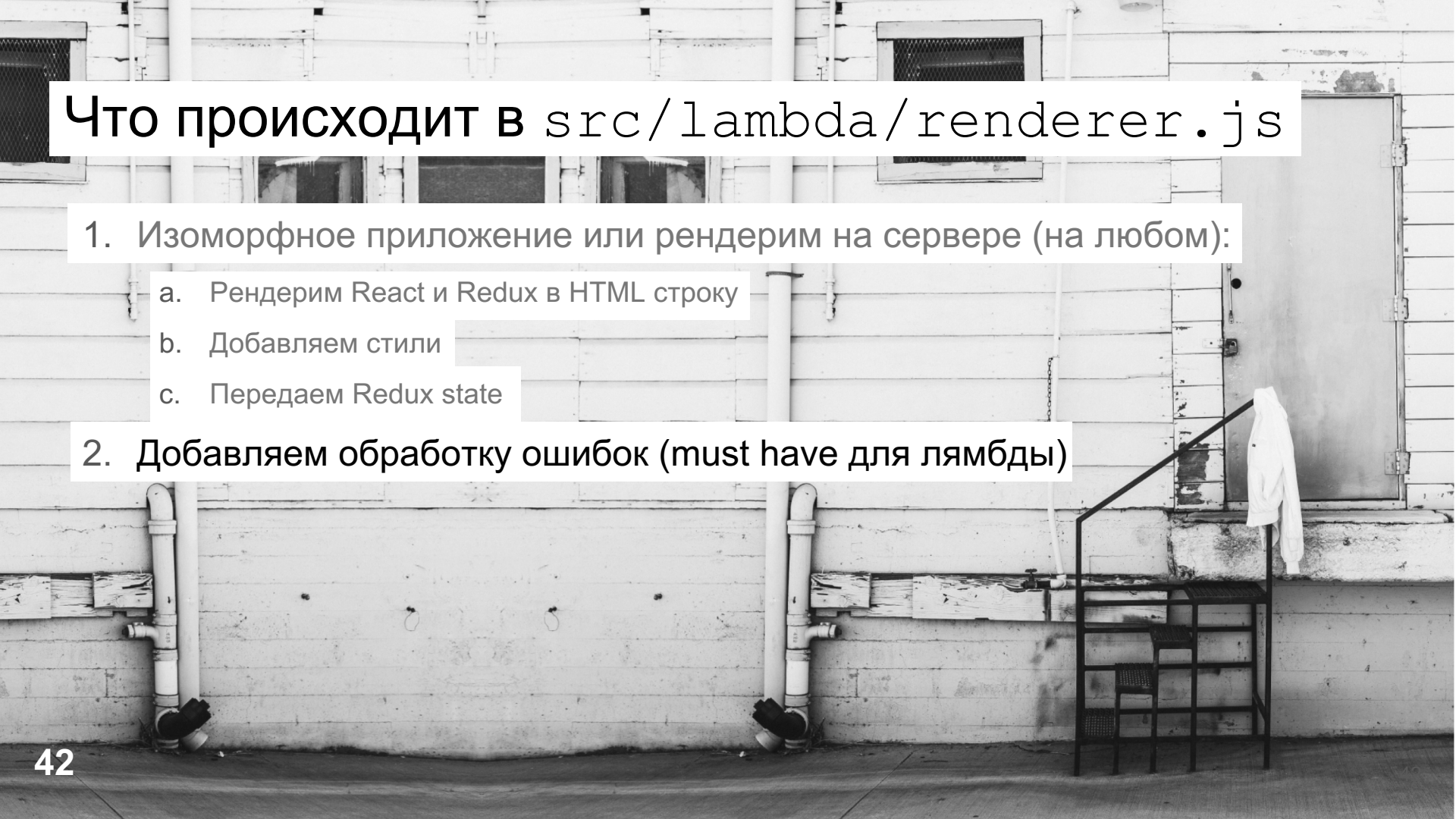
```
1  const styleTag = collectInitial(); // from 'node-style-loader'
2  const generateHTML = (markup, styleTag) => (
3    `<!doctype html>
4    <html lang="en">
5       <head>
6         ${styleTag}
7       </head>
```

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>
```

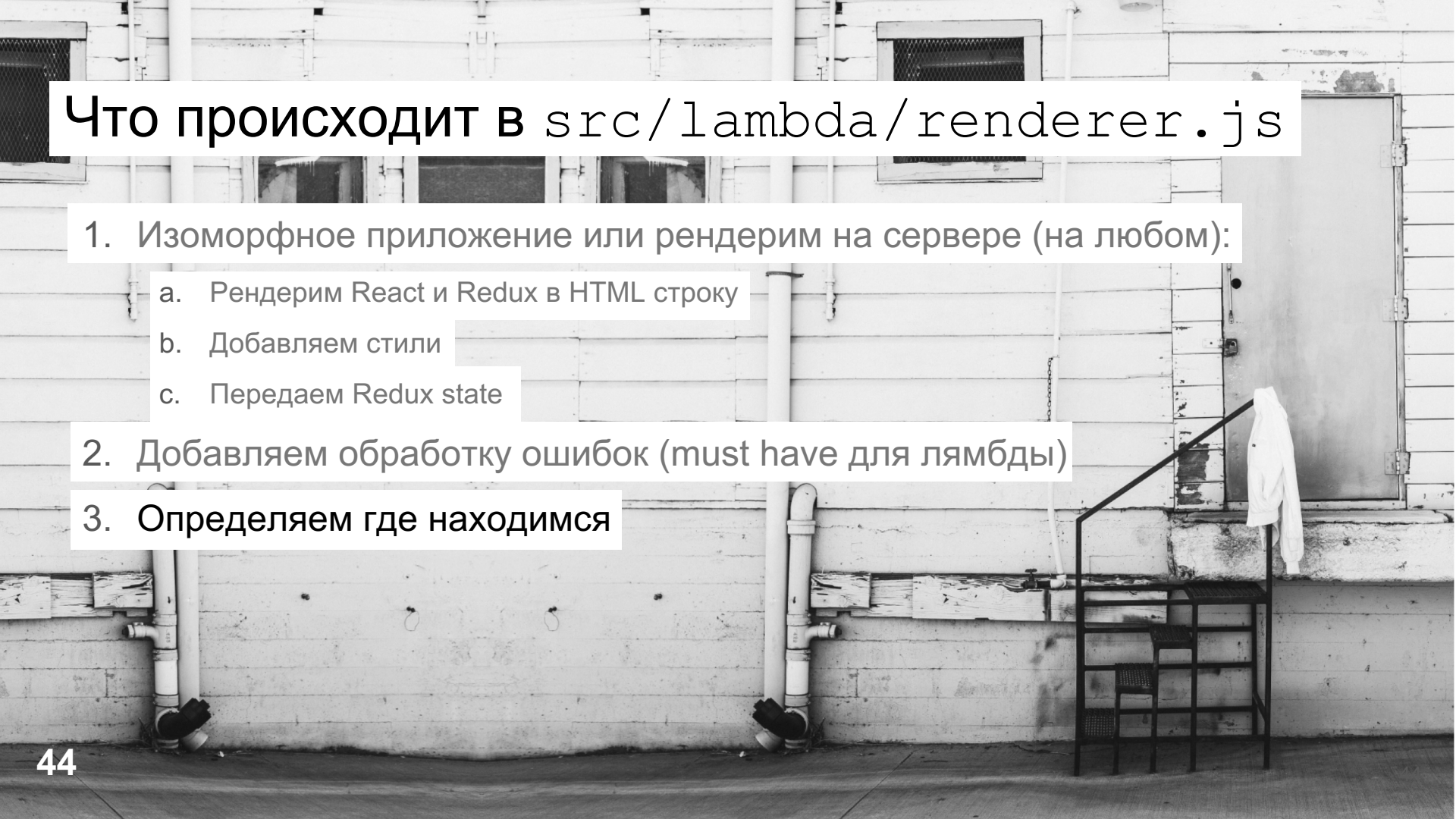


Что происходит в `src/lambda/renderer.js`

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`

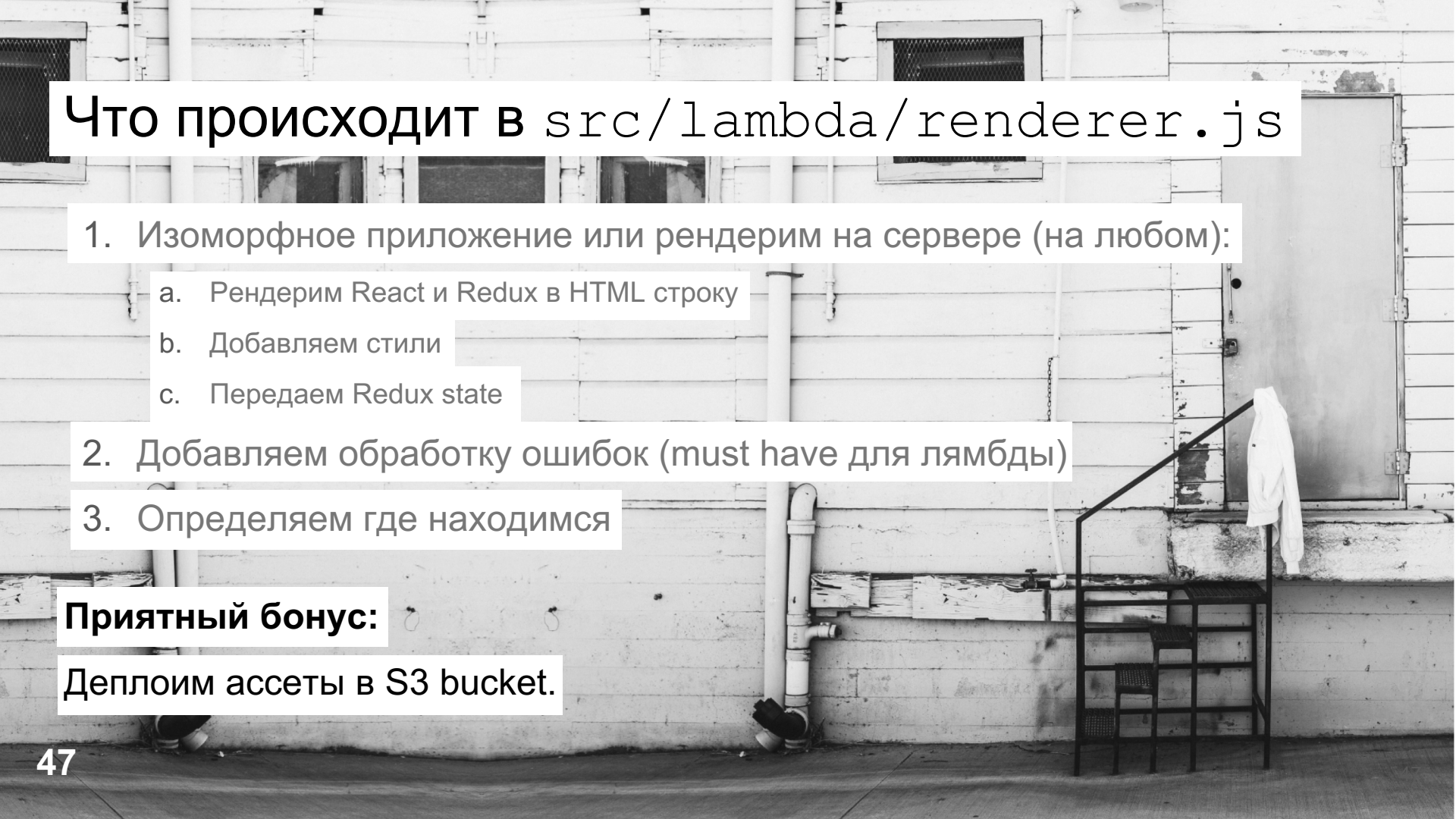
1. Изоморфное приложение или рендерим на сервере (на любом):
    a. Рендерим React и Redux в HTML строку
    b. Добавляем стили
    c. Передаем Redux state
2. Добавляем обработку ошибок (must have для лямбды)
3. Определяем где находимся

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"
...
```



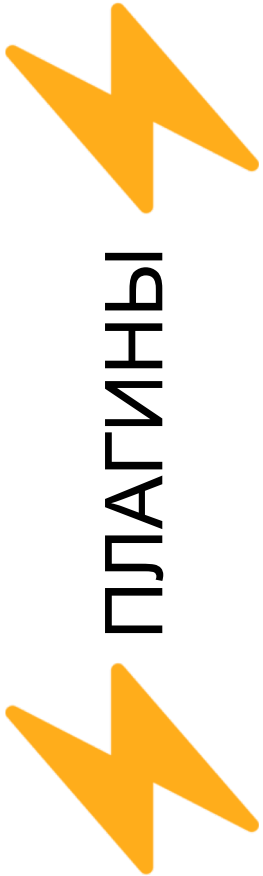
Что происходит в `src/lambda/renderer.js`

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.

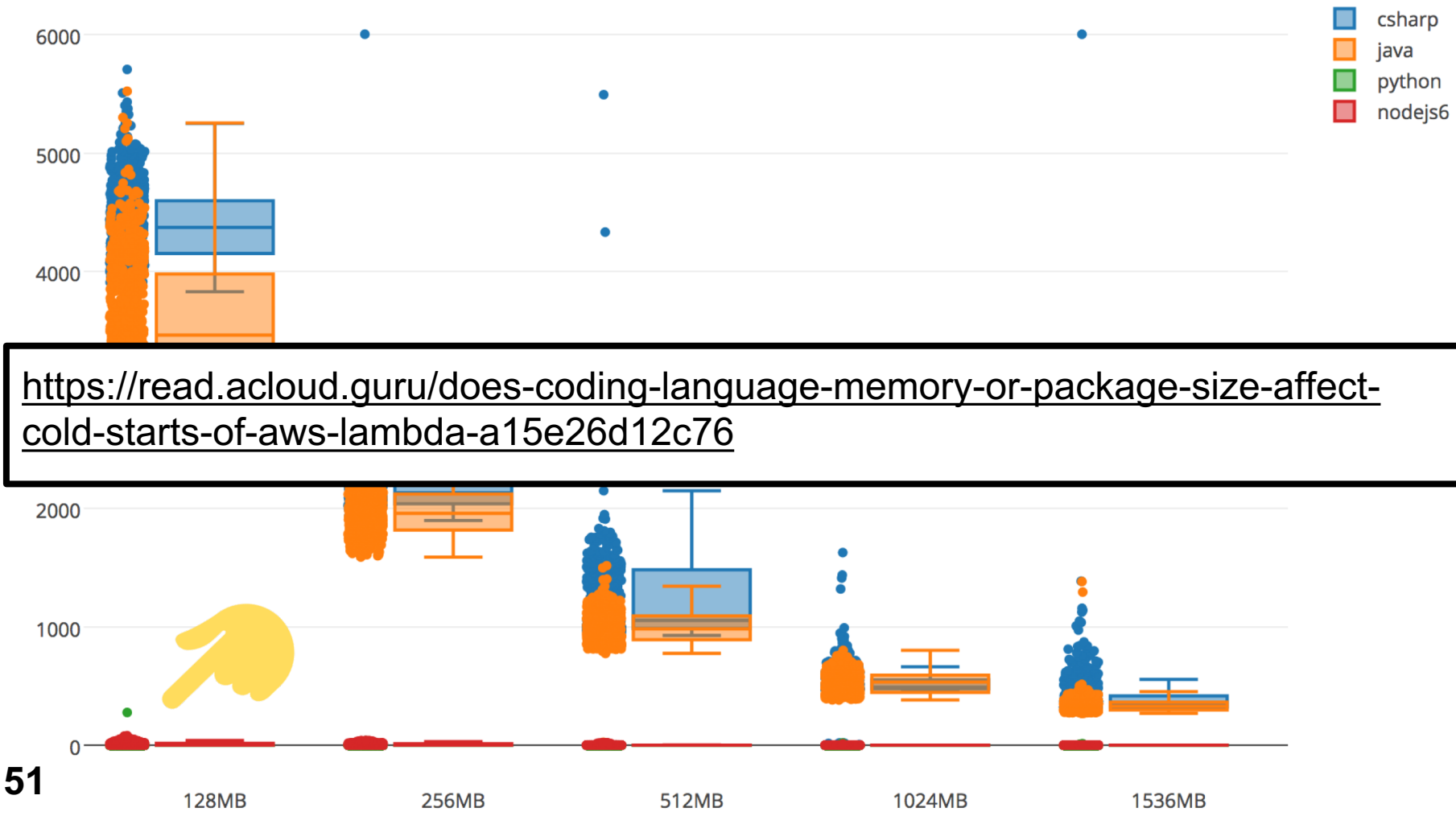
Static assets `serverless.yml`




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

Минусы лямбда функций

Не время холодного старта! (для Node.js)



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

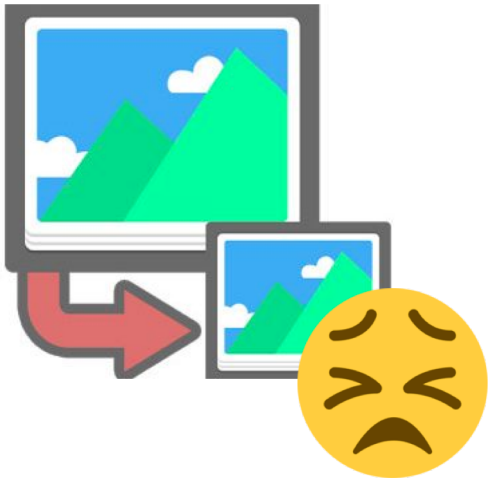
Минусы лямба функций

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

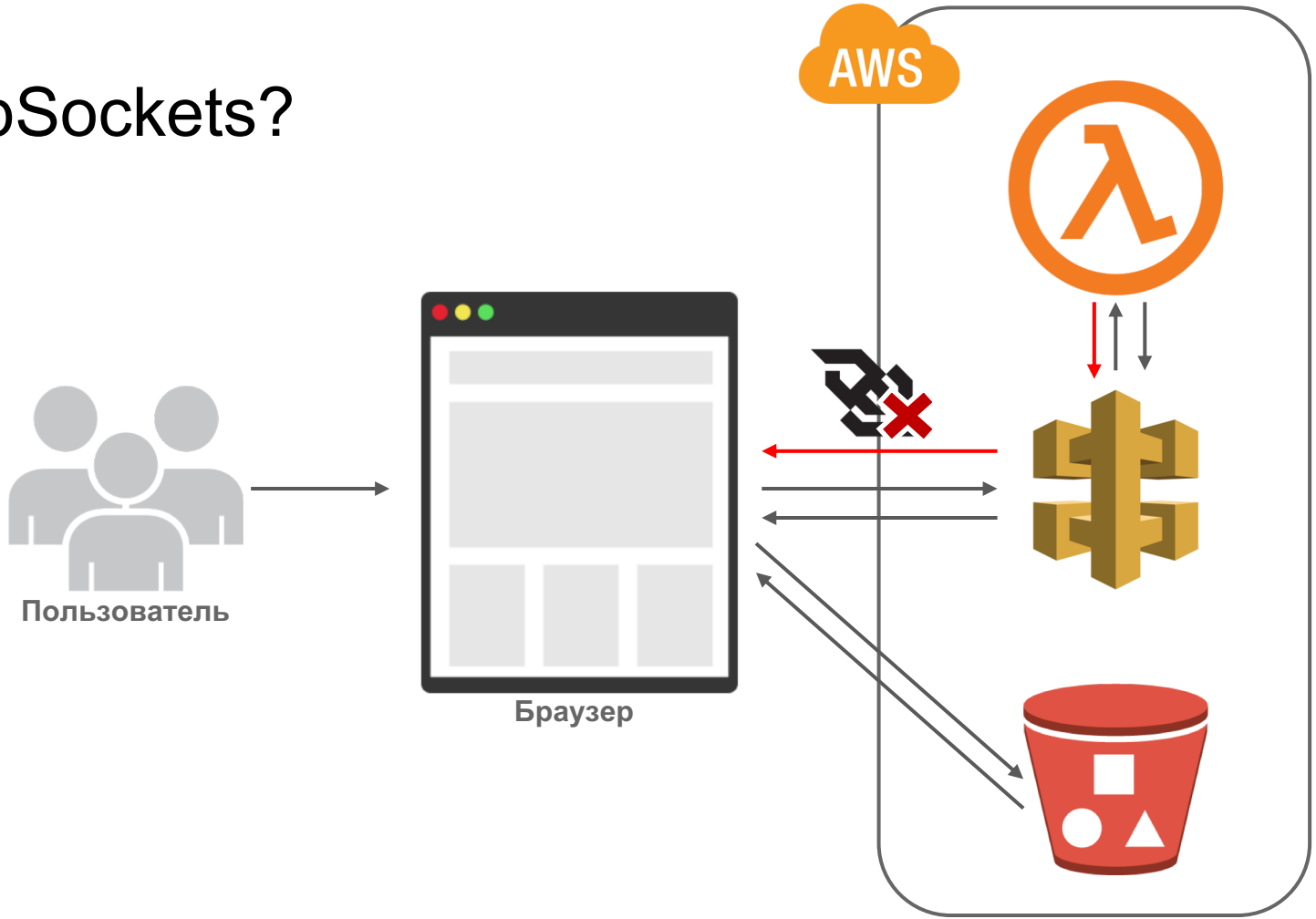
😈 node_modules 😈

Минусы лямба функций

- Ограничения по размеру кода функции
- Ограничения по времени выполнения **(<= 5мин)**



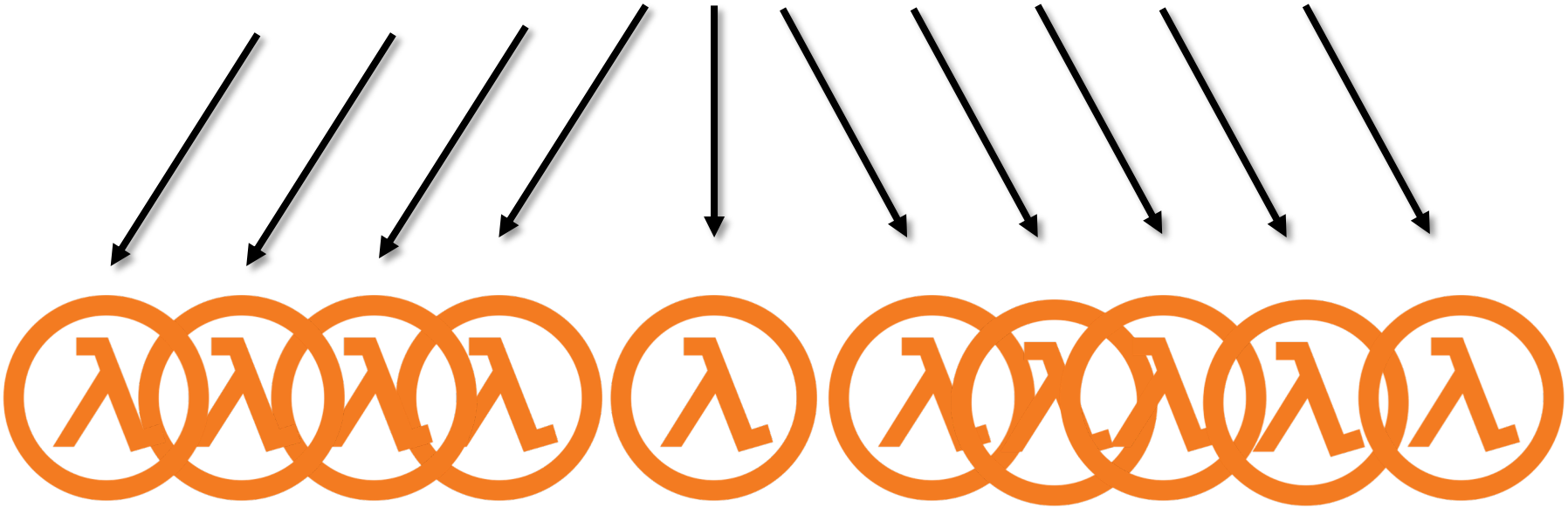
WebSockets?



Минусы лямба функций

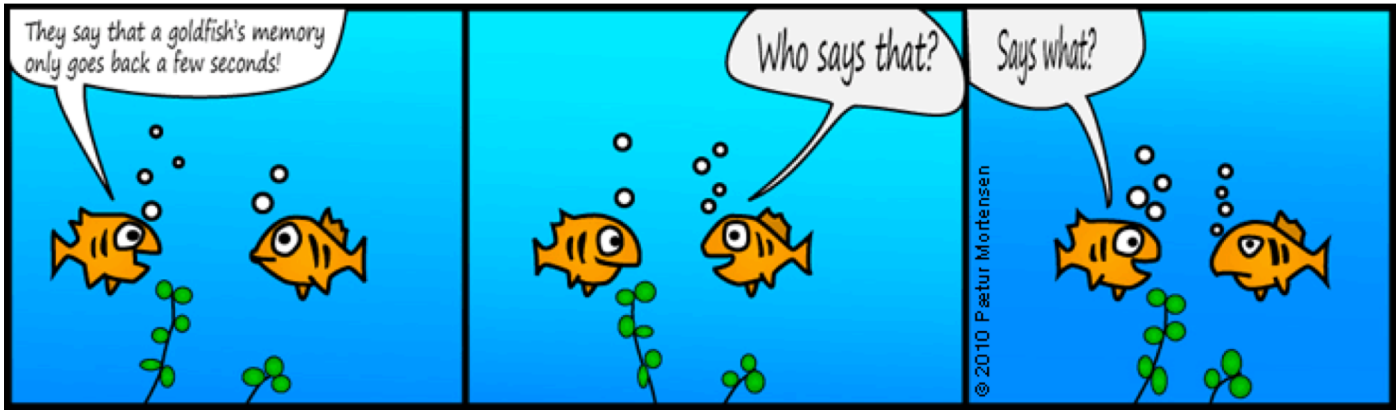
- Ограничения по размеру кода функции
- Ограничения по времени выполнения
- Количество параллельных функций **- 500 - 3000**

Маштабирование лямбды

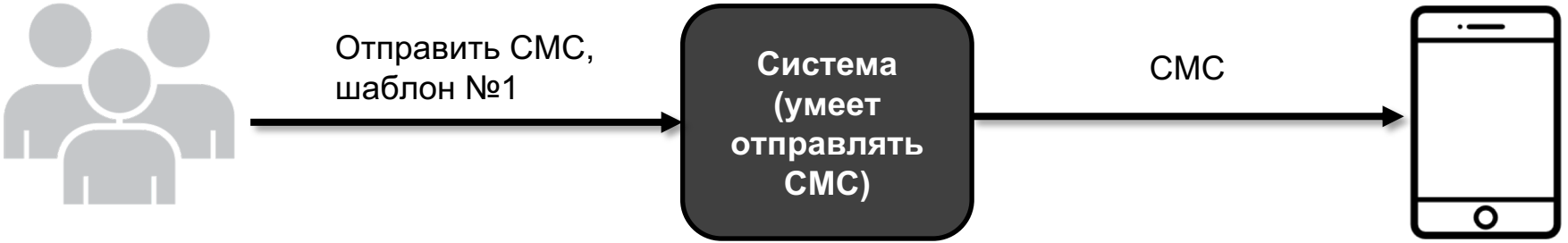


Минусы лямба функций

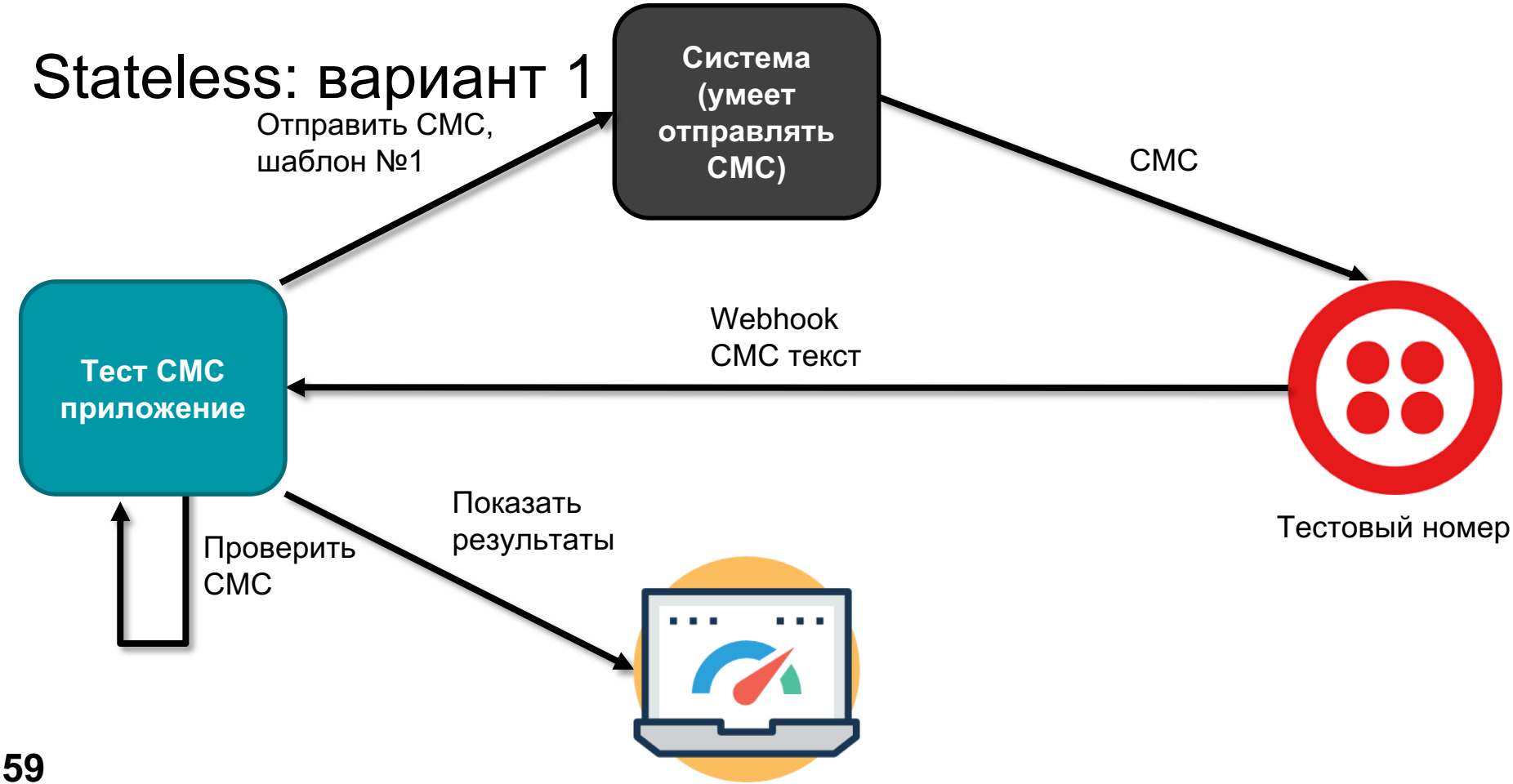
- Ограничения по размеру кода функции
- Ограничения по времени выполнения
- Количество параллельных функций
- Stateless



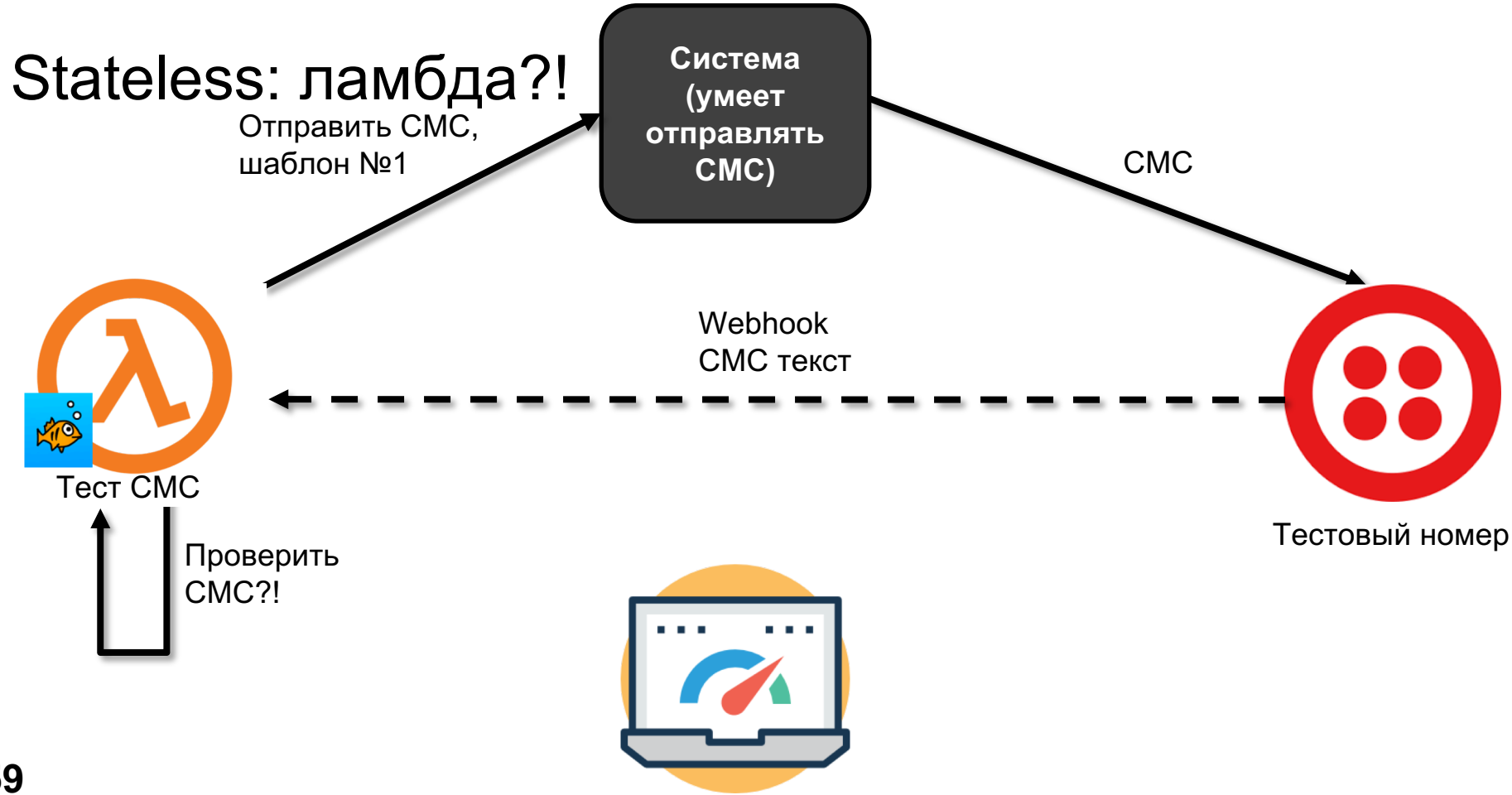
Stateless: Протестировать отправку СМС



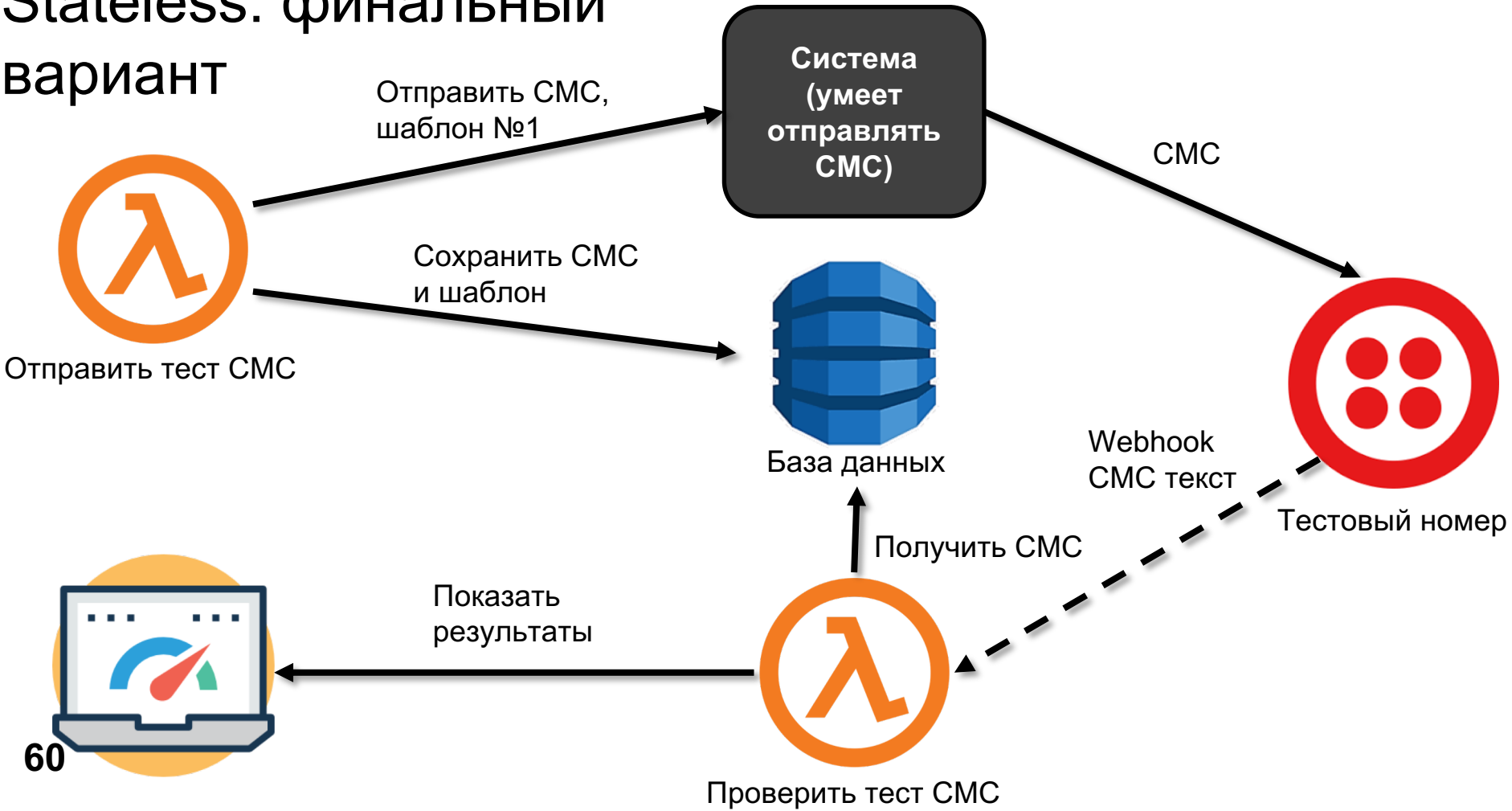
Stateless: вариант 1



Stateless: лямбда?!



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?

Что еще можно сделать на лямбдах?

HTTP services

IoT

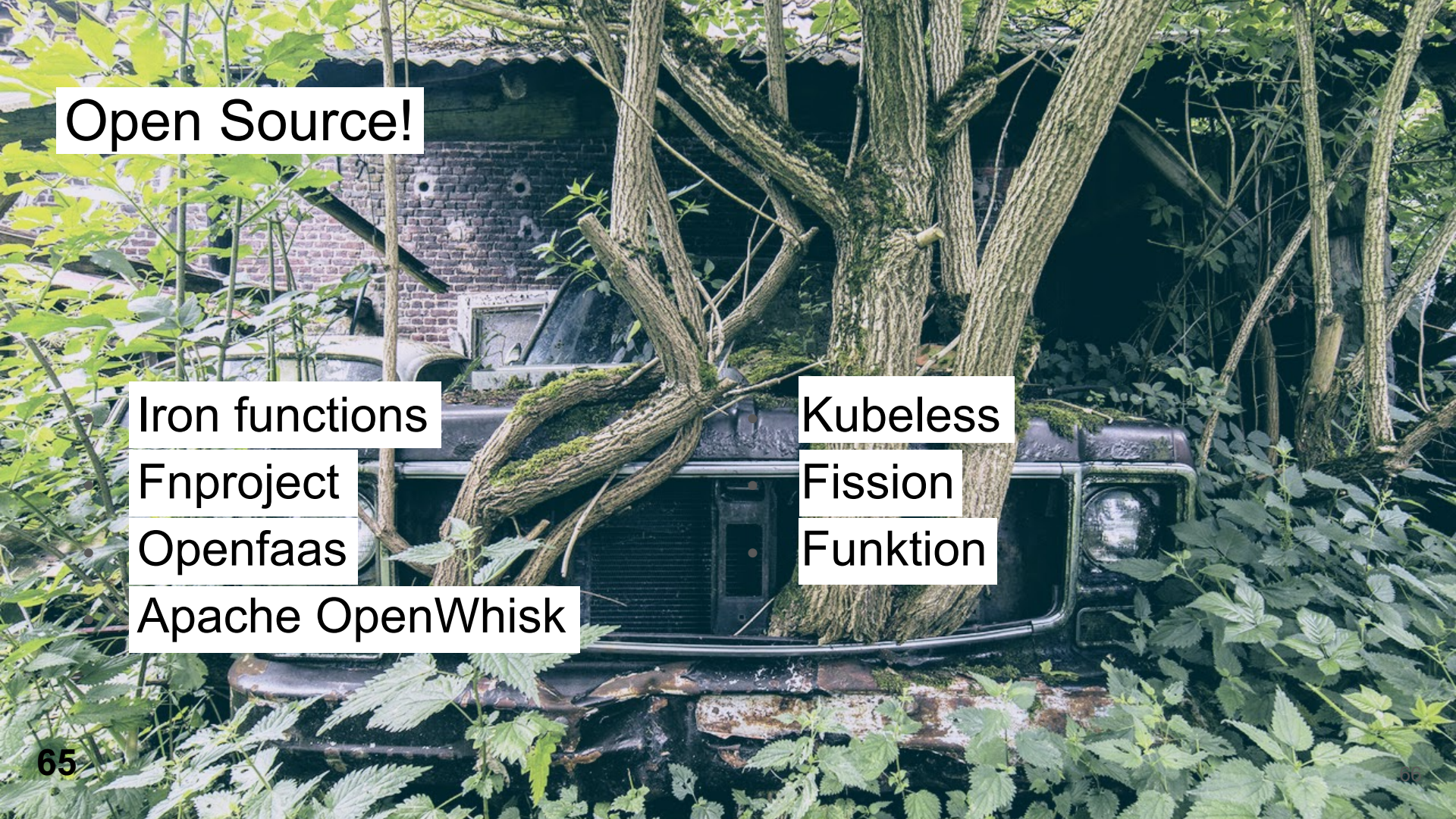
Chat Bots

Image/Video conversions

Machine Learning

Batch Jobs


Есть ли жизнь
после амазон?
Google
ISOLA PALMARIA
Azure
TRAMONTO
BAR
Twillio
IBM


Open Source!
Iron functions
Fnproject
Openfaas
Apache OpenWhisk
Kubeless
Fission
Funktion

Спасибо!

Marina-Miranovich

obnoxious_marie

@obnoxious_mari