

AWS Lambda & API Gateway Overview

The new paradigm in web scale application development

February 2016

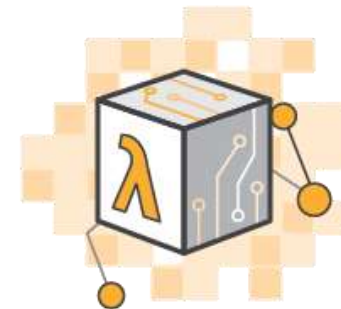
Michael Behrens

CTO, R2AD, LLC

<http://www.R2AD.com>



This briefing is: **UNCLASSIFIED**



Amazon API Gateway

Host the API and
route API calls



AWS Lambda

Execute our app's
business logic



Amazon Cognito

Generate temporary
AWS credentials



Amazon DynamoDB

Data store



The traditional web deployment model

- **Build/Deploy a machine, or more recently a VM**

- Pay for the OS license, maintenance, and security

- **Deploy a web server**

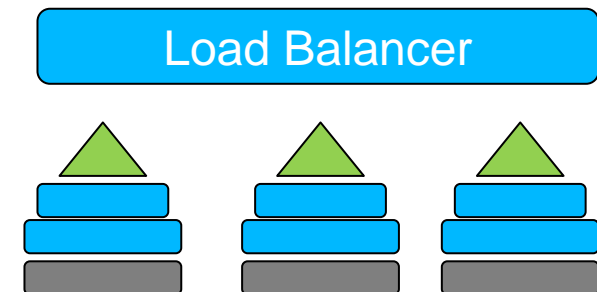
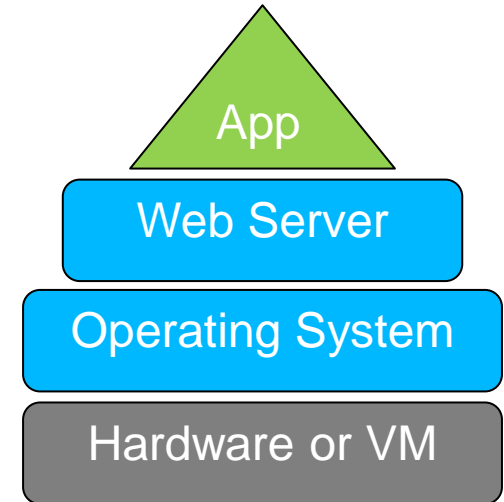
- Pay for the OS license, maintenance, and security

- **Deploy application code that runs on top of that web stack**

- Code gets executed (i.e. Java, JavaScript)

- **Implement ways to make it scalable**

- Scale up, or
- Add a load balancer and scale out by adding more VMs (either statically or dynamically)
- Pay for the additional costs (OSs, VMs) and integration/test



AWS API Gateway Details

- **API Gateway** introduced July 2015
 - “makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale”
 - HTML or ReSTful services

Ref: <https://aws.amazon.com/releasenotes/Amazon-API-Gateway/5820792193066785>

- **Event Driven**
 - Web based management
 - Dashboard – visualization
 - Throttling rules
 - Authorization model



Amazon
Lambda

- **Scalable**

AWS Lambda Details

- **Lambda** introduced November 2014
 - “A compute service that runs your code in response to events and automatically manages the compute resources for you, making it easy to build applications that respond quickly to new information”
 - Support for additional languages are on the roadmap
 - Unknown performance characteristics
 - Need more experiences, studies, proof-of-concepts

Ref: <https://aws.amazon.com/releasenotes/AWS-Lambda/8269001345899110>

- **Triggered by Events**

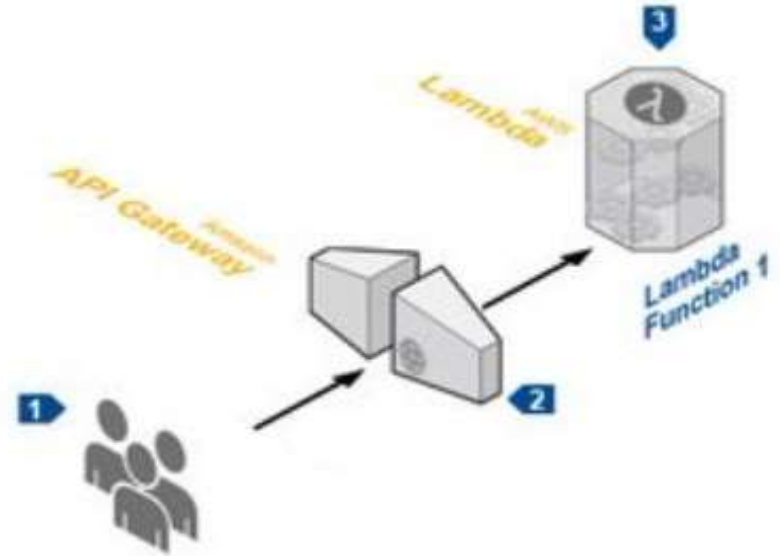
- Periodic
- Items added to an S3 bucket
- API Gateway call



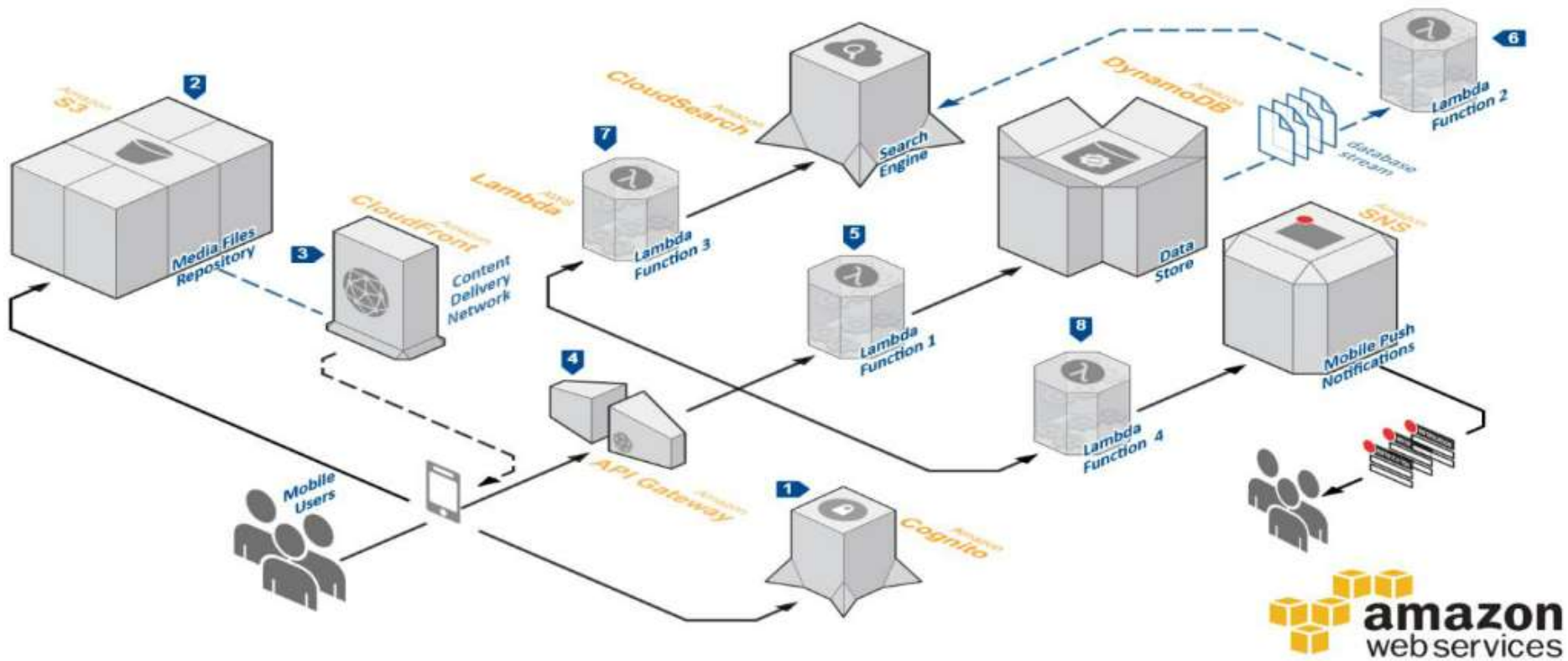
- **Java, JavaScript, Python supported**

Together: a new paradigm for applications

- **Configure an AWS API gateway**
 - Acts as web point of presence
- **Deploy Lambda functions**
 - Code gets executed (i.e. Java, JavaScript)
 - Dashboard Monitor/Log Execution
 - Control Authorization
 - Command line or via the console
- **The entire mechanism is fully scalable by default.**
 - Additional AWS APIs provide support for database services, messaging (i.e. SNS), and static content (i.e. S3).
 - The AWS cloud provides the underpinnings for these higher level services. However that aspect is hidden, abstracted away from development and operations.
- **In many cases, the same code (i.e. Java) can be used.**
 - Essentially, API Gateway & Lambda provide a scalable web platform



AWS Serverless Architecture Example

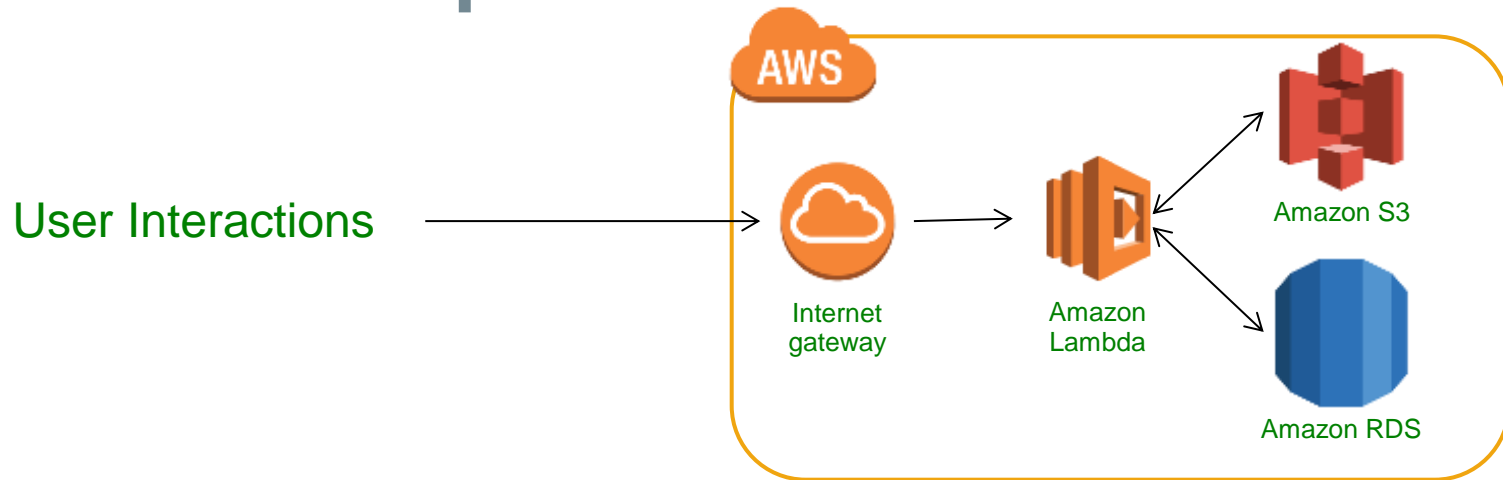


Source: <https://s3.amazonaws.com/awslambda-reference-architectures/mobile-backend/lambda-refarch-mobilebackend.pdf>

AWS API Gateway + Lambda Benefits

- **No Servers to Manage**
 - Simple. Write code and deploy it.
 - SDK supports most workflows
 - Microservice without servers
- **Scale automatically based on events**
- **Reduced Cost**
 - Cost of Lambda & API Gateway is cheaper
 - When compared with the cost of EC2 instances + OS licenses + web server licenses
 - Charged for every 100ms your code executes and the number of times code is triggered

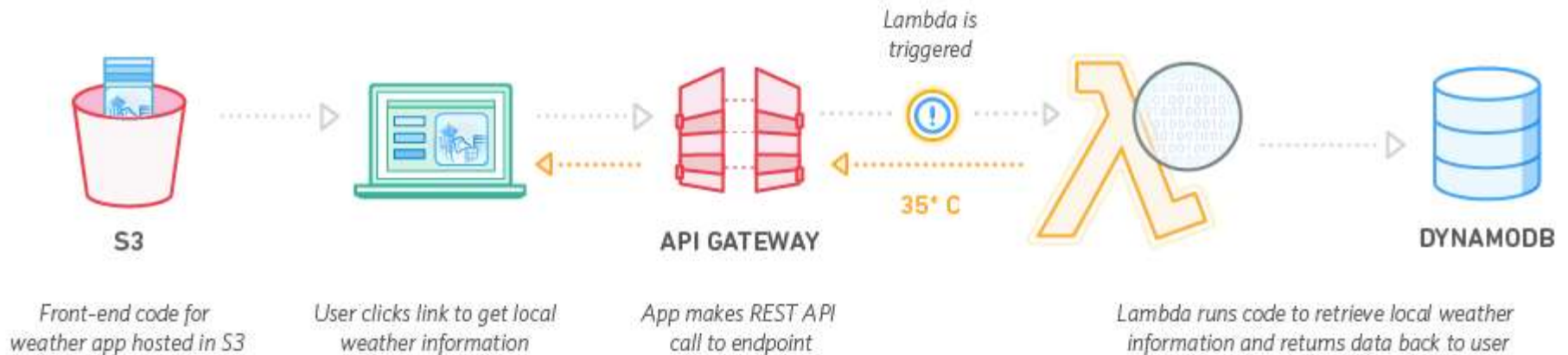
Sample CRUD workflow



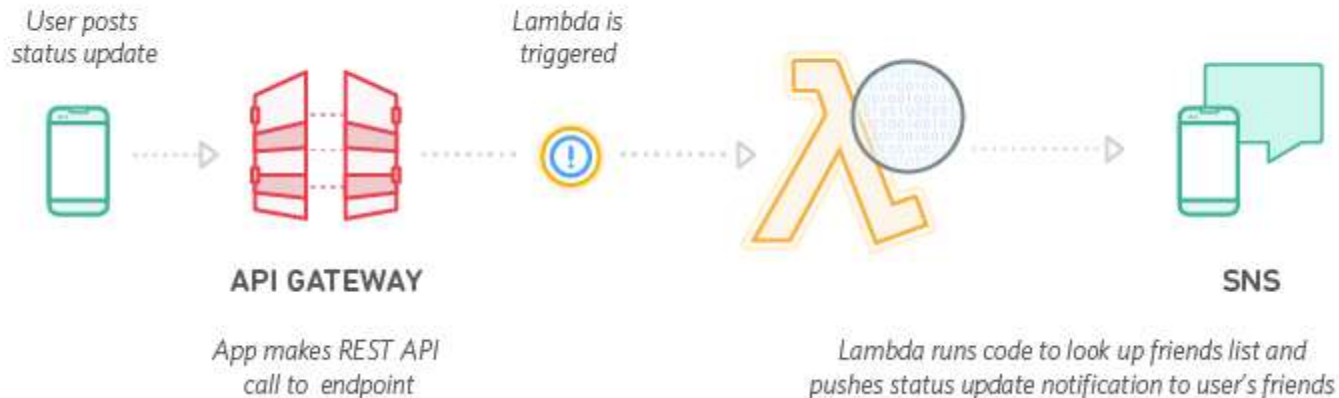
- HTTP endpoint hits the API gateway
- API Gateway then invokes the configured AWS Lambda function
- The Lambda function connects with storage services to create, replace, update, or delete data
 - HTML response or JSON, XML, etc.

Another Examples

Example: Weather Application



Example: Mobile Backend for Social Media App




Demo

- A live demonstration of a simple web application
 - Installed and running on AWS Cloud using API Gateway and Lambda functions
- AWS Demo code available on internet
 - <https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started-mappings.html>
 - <http://www.serverless.com/>
 - <https://github.com/aws-labs/lambda-refarch-mobilebackend>
- R2AD is working to apply this technology to applications this fiscal year
 - Proof of concept to reduce sustainment cost and increase scalability & maintainability
 - If successful, deployment in 2017
 - AWS GovCloud first, then later in C2S

- serverless project create

```
E:\behrens\programs\CloudStudy\Lambda\serverless>serverless project create
```



```
The Serverless Application Framework  
serverless.com, v0.4.2
```

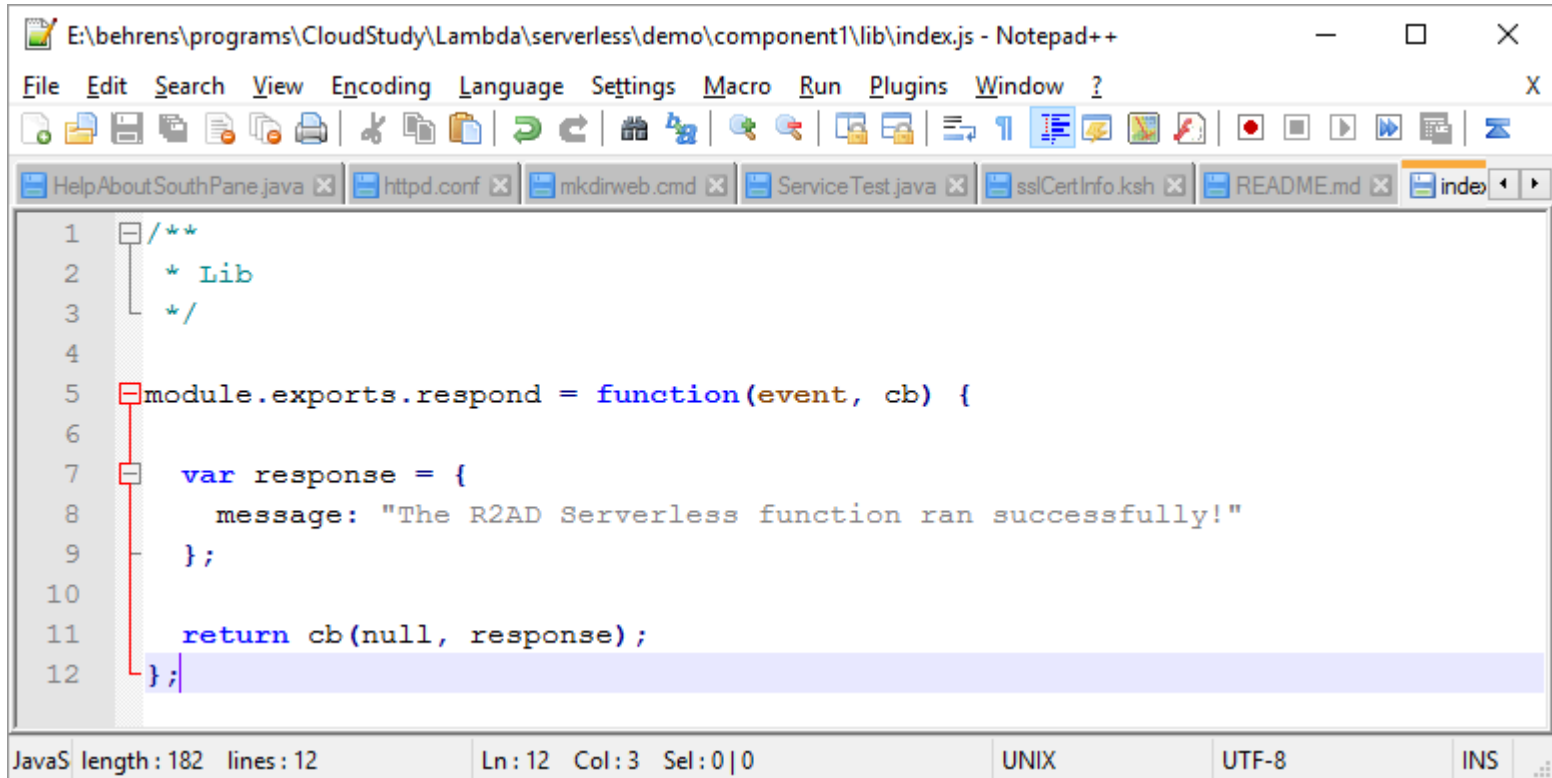
```
Serverless: Initializing Serverless Project...  
Serverless: Enter a name for this project: <serverless-nkxq3g> demo  
Serverless: Enter a universally unique project bucket name: <serverless-nkxq3g> lambdaservice  
Serverless: Enter an email to use for AWS alarms: <me@serverless-nkxq3g.com>  
Serverless: Select a region for your project:  
> us-east-1  
us-west-2  
eu-west-1  
ap-northeast-1  
Serverless: Select an AWS profile for your project:  
> default  
Serverless: Creating stage "dev"...  
Serverless: Creating region "us-east-1" in stage "dev"...  
Serverless: Creating your project bucket on S3: serverless.us-east-1.lambdaservice...  
Serverless: Deploying resources to stage "dev" in region "us-east-1" via Cloudformation (~3 minutes)...  
Serverless: Successfully deployed "dev" resources to "us-east-1"  
Serverless: Successfully created region "us-east-1" within stage "dev"  
Serverless: Successfully created stage "dev"  
Serverless: Successfully initialized project "demo"
```

```
E:\behrens\programs\CloudStudy\Lambda\serverless>
```

Ref: <http://docs.serverless.com/docs/installing-serverless>

Edit the function and re-deploy

- The function can be locally edited, in this case, and re-published by re-executing the deploy command.



```
E:\behrens\programs\CloudStudy\Lambda\serverless\demo\component1\lib\index.js - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
HelpAboutSouthPane.java httpd.conf mkdirweb.cmd ServiceTest.java sslCertInfo.ksh README.md index
1  /**
2   * Lib
3   */
4
5  module.exports.respond = function(event, cb) {
6
7      var response = {
8          message: "The R2AD Serverless function ran successfully!"
9      };
10
11     return cb(null, response);
12 };
```

JavaS length: 182 lines: 12 Ln: 12 Col: 3 Sel: 0|0 UNIX UTF-8 INS







- In the browser, visit:
 - <https://vpssc5pe0i8.execute-api.us-east-1.amazonaws.com/dev/function1>

Examine the AWS API endpoint

The screenshot displays the AWS API Gateway console interface. At the top, navigation tabs include AWS, Services, S3, EC2, IAM, and Edit. The user is logged in as Michael Behrens in the N. Virginia region. The main navigation bar shows 'Amazon API Gateway', 'APIs', 'demo', and 'Resources'. The 'Resources' section on the left lists the root resource '/' and its child resource 'GET'. The 'Deploy API' button is visible. The main content area is titled '/ - GET - Method Execution' and includes a 'Delete Method' button. The execution flow is shown as follows:

- Client** (represented by a lightning bolt icon) sends a request to the **Method Request** box.
- The **Method Request** box contains:
 - Auth: NONE
 - ARN: arn:aws:execute-api:us-east-1:442496353009:vp5c5pe0
- The request is then sent to the **Integration Request** box, which contains:
 - Type: LAMBDA
 - Region: us-east-1
- The request is then sent to the **Lambda demo-component1-function1** function.
- The function returns an **Integration Response** box, which contains:
 - HTTP status pattern: (dropdown menu)
 - Output passthrough: Yes
- The response is then sent to the **Method Response** box, which contains:
 - HTTP Status: 200
 - Models: application/json => Empty
- The response is finally sent back to the **Client**.

Examine the AWS Lambda Function

 **AWS** ▾ **Services** ▾  **S3**  **EC2**  **IAM**  **API Gateway**  **Lambda** **Edit** ▾ **Michael Behrens** ▾ **N. Virginia** ▾ **Support** ▾

Lambda > **Functions** > **demo-component1-function1** ARN - arn:aws:lambda:us-east-1:442496353009:function:demo-component1-function1

Test **Actions** ▾

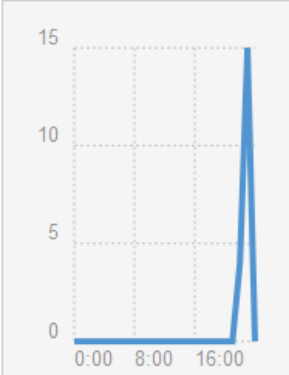
Versions [show aliases](#)

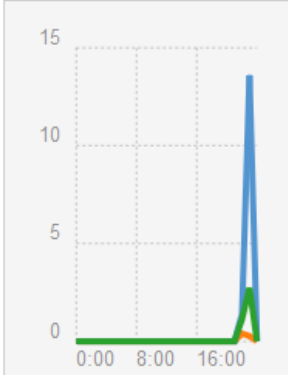
Filter

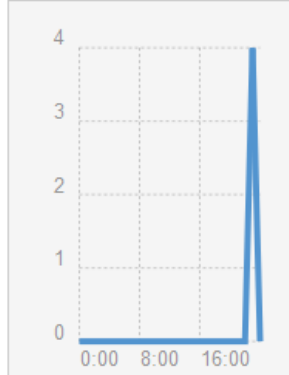
- \$LATEST** (2016/02/24)
Serverless Lambda function for project: d...
dev
- 9 (2016/02/24)
Serverless Lambda function for project: d...
- 8 (2016/02/24)
Serverless Lambda function for project: d...
- 7 (2016/02/24)
Serverless Lambda function for project: d...
- 6 (2016/02/24)
Serverless Lambda function for project: d...
- 5 (2016/02/24)
Serverless Lambda function for project: d...


Code **Configuration** **Event sources** **API endpoints** **Monitoring** ?

CloudWatch metrics at a glance (last 24 hours) [View logs in CloudWatch](#)

Invocations ↺


Duration ↺


Errors ↺


Throttles ↺


Other monitoring information

Last modified date 2016-02-25T04:17:02.033+0000

UNCLASSIFIED

