Serverless

and AWS lambda

https://www.egnyte.com/blog/2011/11/the-file-server-is-dead/

No server is easier to manage than no server.

Werner Vogels, CTO, Amazon.com

BaaS

Backend as a Service - rich client apps (think SPAa, Mobile Apps) that rely mostly or entirely on 3rd party applications / services in the cloud (2011 on)

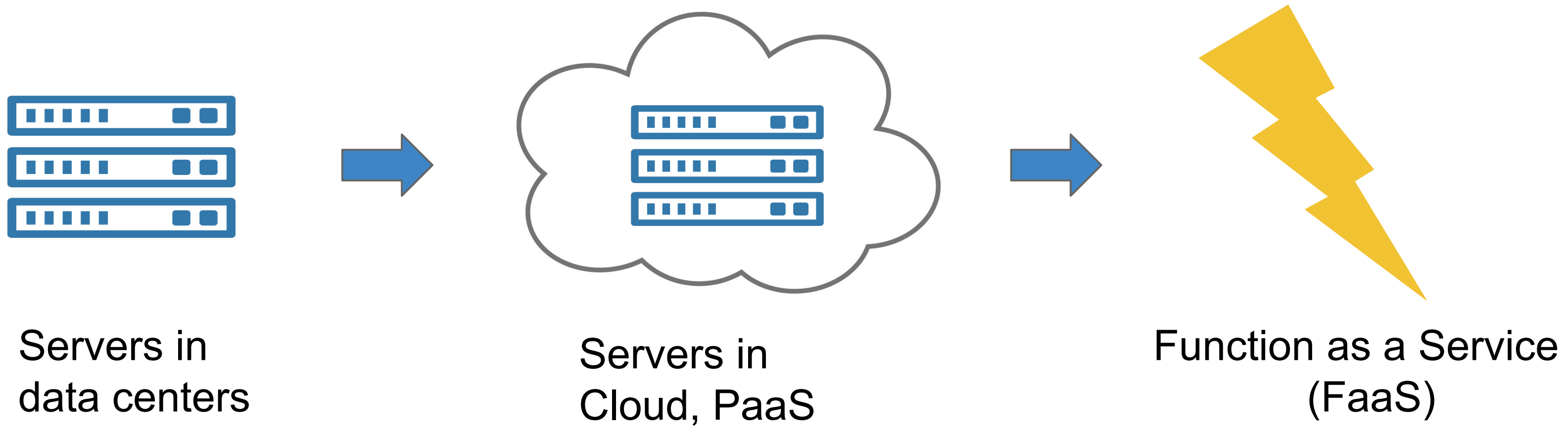
FaaS - paradigm shift in cloud

Function as a Service - functions that run in stateless compute containers that are event-triggered, short lived, and fully managed by a 3rd party (2014 on)

Serverless == FaaS

throughout the rest of the presentation

How did we get here?


Servers in data centers
Servers in Cloud, PaaS
Function as a Service (FaaS)

PaaS

Wrong unit of abstractions: Deployed (Monolithic) Applications

Serverless

Services and Functions are the platform abstractions and unit of deployment

Why Serverless?

- Scalability
- Costs scale per request
- Push based, event driven pipelines
- Security - less time availability in one invocation
- Fault tolerance - independent functions, doesn’t affect others
- No OS config or security patching

Serverless (FaaS) Providers

AWS Lambda

Google Cloud Platform

Azure Functions

Webtask

Iron.io

IBM OpenWhisk

Full list here

- Zimki - first "pay as you go" code execution platform, 2006
- Google App Engine (metered billing, no arbitrary code), 2008
- **AWS Lambda, introduced by Amazon in 2014**

AWS Lambda

What is a lambda function?

- unit of work (your code)
- responds to individual requests and events
- stateless
- scales based on requests and events

  (no risk of over or under provisioning)

Use cases

APIs; GraphQL

Mobile Apps

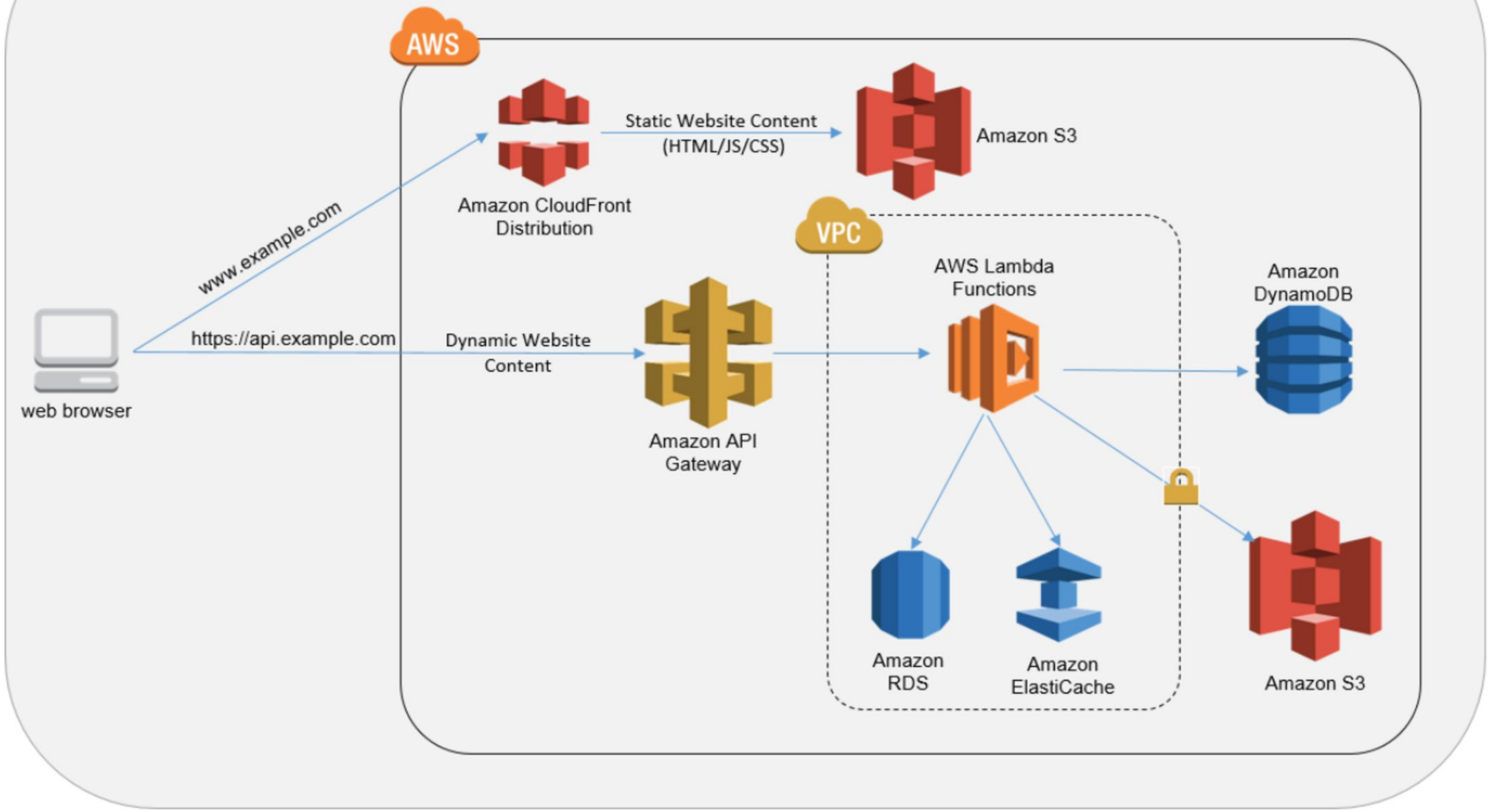
IoT

Data Analysis; perform actions upon data ingestion; avoids polling (~ Pub/Sub)

Operation tasks: alarms, scheduled jobs, scheduling snapshots; cleanup

Microservices (serverless ~ nanoservices)

Amazon S3 Hosted Websites

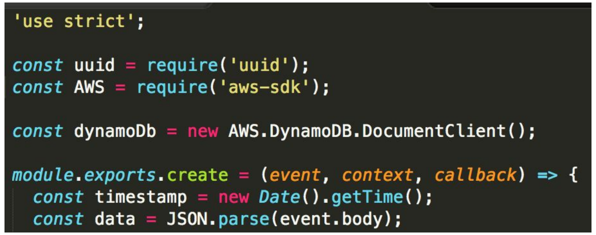


https://github.com/jonascheng/serverless-architecture-with-aws

Always use lambdas?

- Limited to 5 minutes per run - not suitable for long running tasks
- Cold start (initialization phase in container)

  Minimize the code outside of the function
  Make package as small as possible
  Remove unnecessary dependencies so the download from s3 happens asap
  Schedule with CloudWatch to invocate function for warmth



```
'use strict';

const uuid = require('uuid');
const AWS = require('aws-sdk');

const dynamoDb = new AWS.DynamoDB.DocumentClient();

module.exports.create = (event, context, callback) => {
  const timestamp = new Date().getTime();
  const data = JSON.parse(event.body);
```

- High dependency on one service (S3) - don’t put all your eggs in one ~~bucket~~ basket? Especially if you remember Feb 2017

Servers are dead, they just don't know it yet

Hacking time!

1. Setup & Hello World (together)
2. S3, IAM Roles, Endpoints (all you!)
3. Events (also you)

Final code available here https://gitlab.com/luiza-salantiu/serverless-demo

1. Setup & Hello World

- ## AWS Lambda

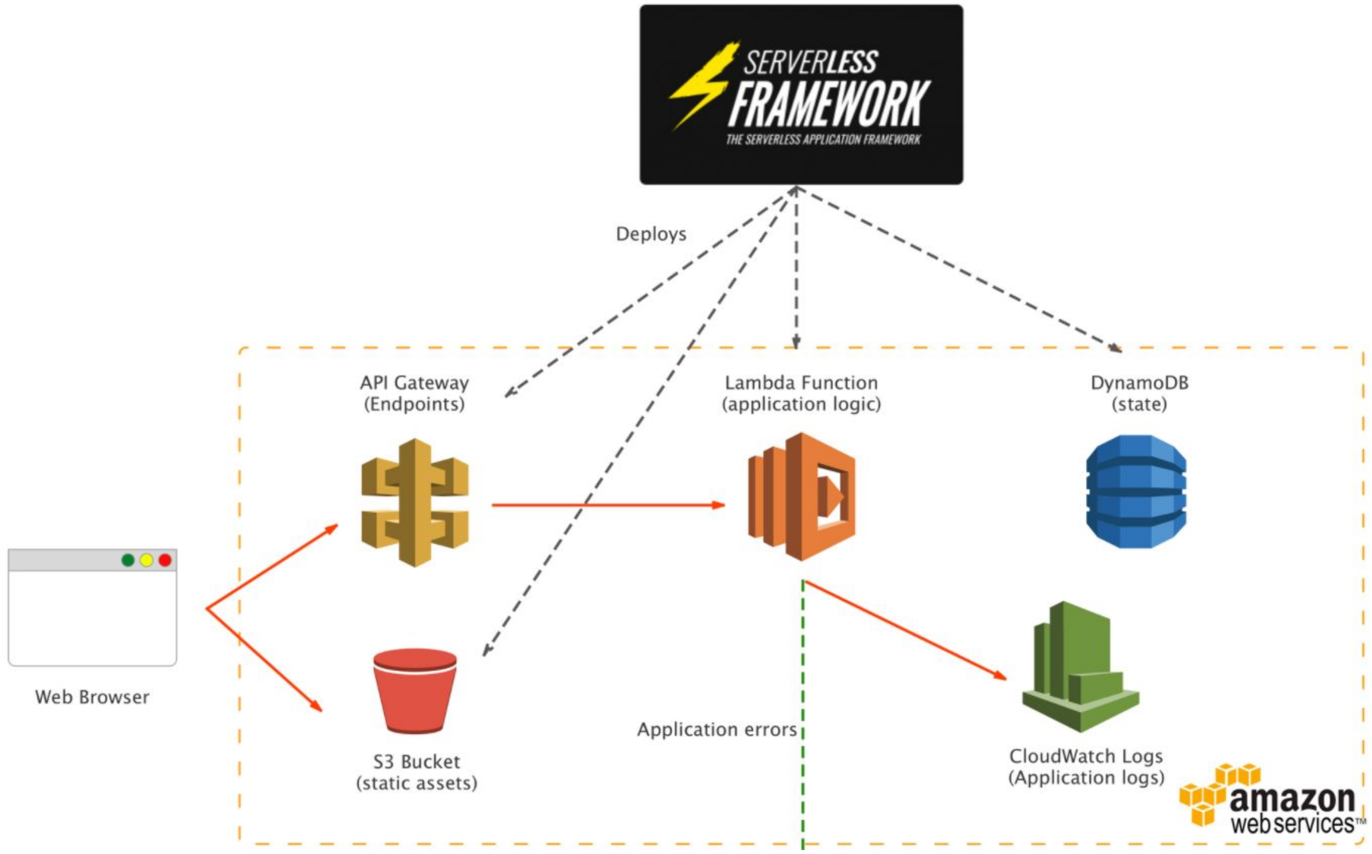
  Hosting and code execution in the cloud

- ## Serverless framework

  https://serverless.com/

  Takes care of defining and deploying infrastructure resources, as well as function code

  no support for ruby yet :(we will use node.js and javascript for the demos



https://rollbar.com/blog/how-gorillastack-used-rollbar-to-level-up/

Setup

1. Install nodejs and npm

Using homebrew

```
$ brew install node
```

How to install with Homebrew

Or from official page

2. Install Serverless Framework (CLI)

```
$ npm install -g serverless
```

Setup

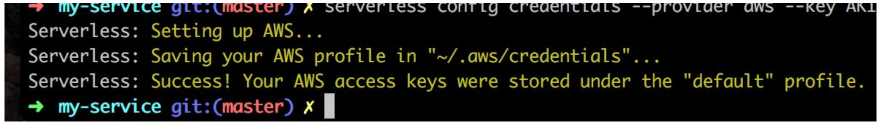
3. AWS Account

If you don't already have one, you can sign up for a free trial that includes 1 million free Lambda requests per month

4. Set AWS Credentials locally

```
$ serverless config credentials --provider aws --key xxx --secret zzz
```

You can also use aws cli or the serverless dashboard to set these



```
➜ my-service git:(master) ✗ serverless config credentials --provider aws --key AKI
Serverless: Setting up AWS...
Serverless: Saving your AWS profile in "~/.aws/credentials"...
Serverless: Success! Your AWS access keys were stored under the "default" profile.
➜ my-service git:(master) ✗
```

If you don’t have your own AWS Account

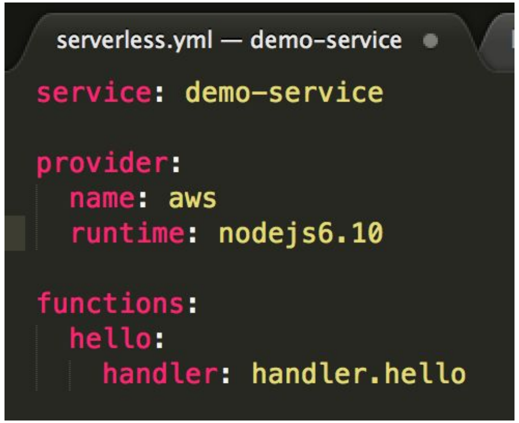
Prefix anything that goes into AWS:

- service name
- bucket name
- notification name

with your name to avoid name collisions.

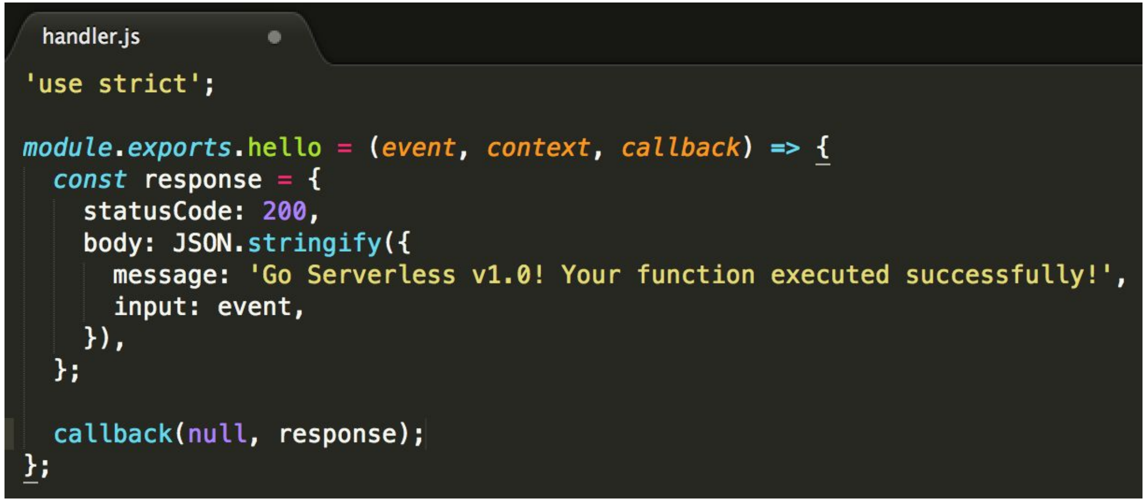
Create serverless service

```
$ serverless create --template aws-nodejs --path <service-name>
```



serverless.yml — demo-service

```yaml
service: demo-service

provider:
  name: aws
  runtime: nodejs6.10

functions:
  hello:
    handler: handler.hello
```

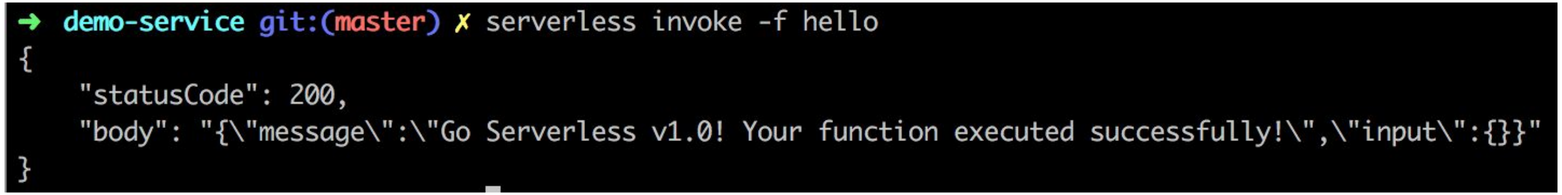


handler.js

```javascript
'use strict';

module.exports.hello = (event, context, callback) => {
  const response = {
    statusCode: 200,
    body: JSON.stringify({
      message: 'Go Serverless v1.0! Your function executed successfully!',
      input: event,
    }),
  };

  callback(null, response);
};
```

Deploy & invoke service

```
$ cd <service-name>

$ serverless deploy

$ serverless invoke -f hello
```

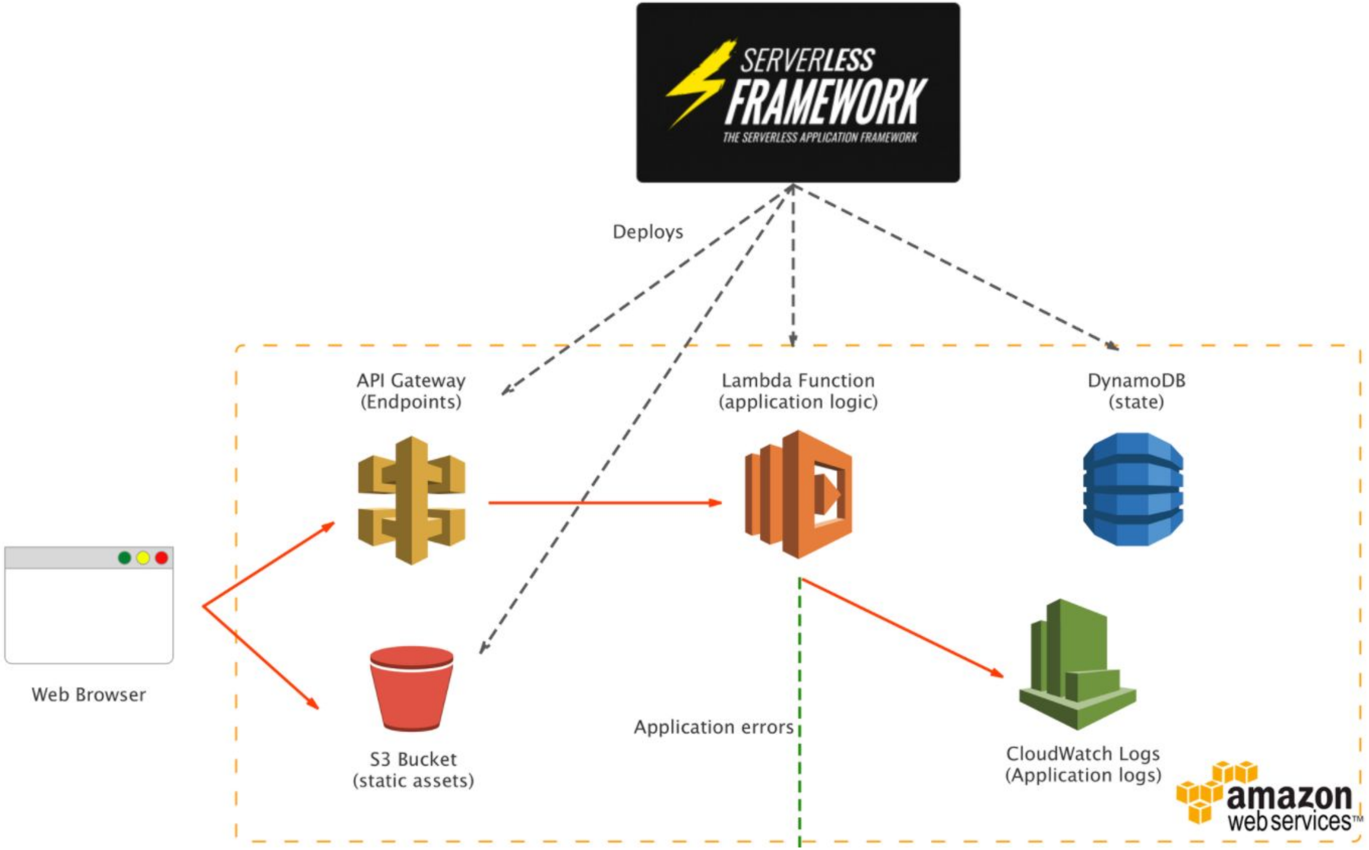


```
➜  demo-service git:(master) ✗ serverless invoke -f hello
{
    "statusCode": 200,
    "body": "{\"message\":\"Go Serverless v1.0! Your function executed successfully!\",\"input\":{}}"
}
```

```
# create new serverless service/project
$ serverless create --template ... --path ...

# deploy verbose mode
$ serverless deploy -v

# deploy single function; recommended
$ serverless deploy function -f hello

# invoke with logs
$ serverless invoke -f hello -l

# invoke locally
$ serverless invoke local -f hello

# trail logs
$ serverless logs -f hello -t

# remove service
$ serverless remove
```

2. S3, IAM Roles, Endpoints

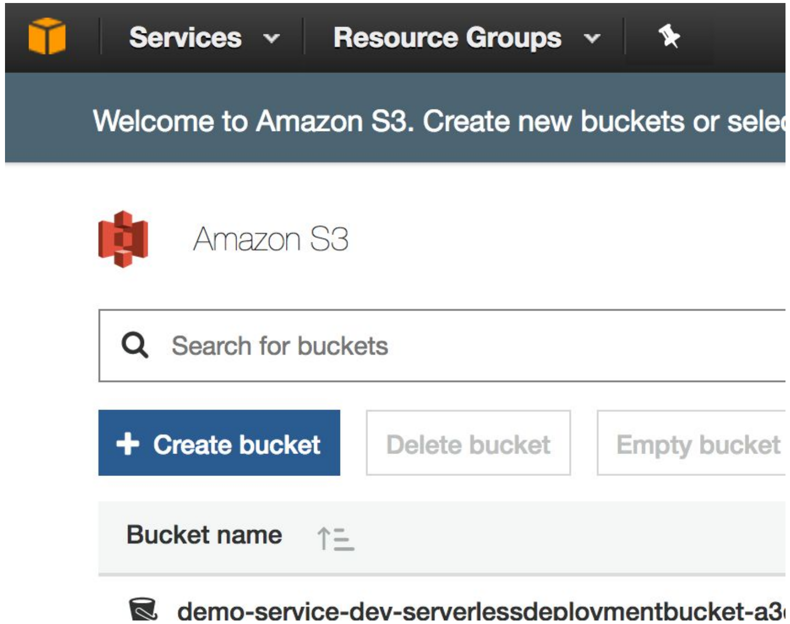
Update Hello World function to show contents from an S3 bucket via HTTP Endpoint


SERVERLESS FRAMEWORK
THE SERVERLESS APPLICATION FRAMEWORK
Deploys
API Gateway (Endpoints)
Lambda Function (application logic)
DynamoDB (state)
Web Browser
S3 Bucket (static assets)
Application errors
CloudWatch Logs (Application logs)
amazon web services™

https://rollbar.com/blog/how-gorillastack-used-rollbar-to-level-up/

Read files from S3

- Create Bucket in S3
  (Services -> Storage -> S3)
- Upload two small images to it

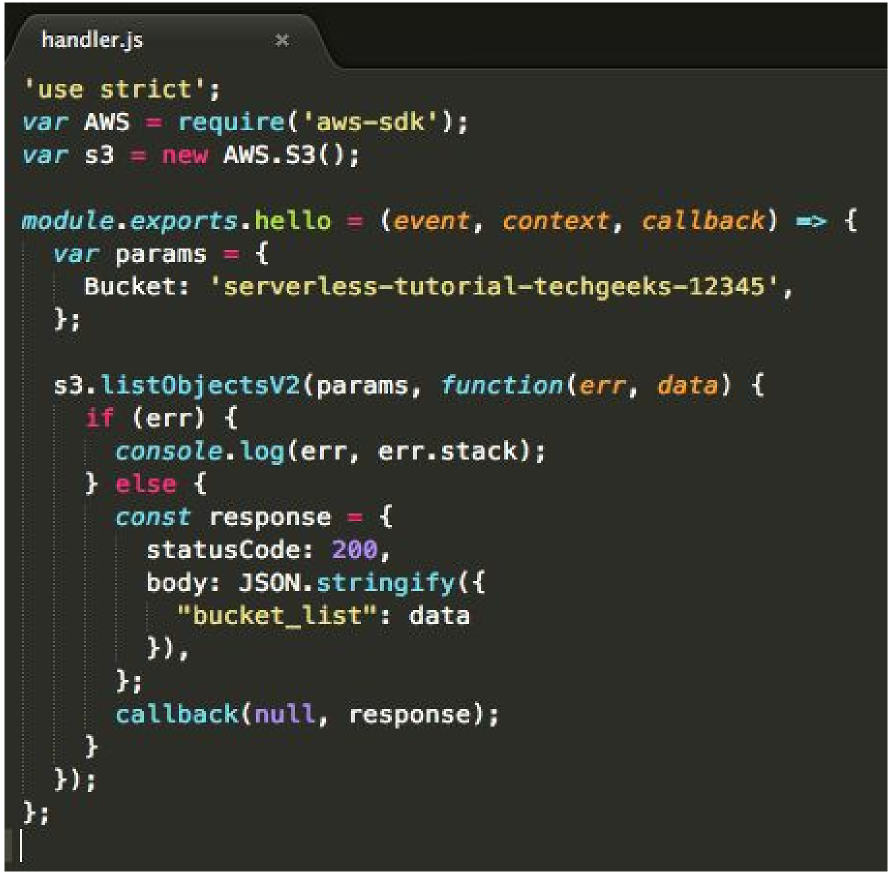

Services
Resource Groups
Welcome to Amazon S3. Create new buckets or selec
Amazon S3

Bucket name
demo-service-dev-serverlessdeploymentbucket-a3

Read files from S3

Update handler.js code to read the data from the bucket you just created

* replace bucket name with yours



handler.js

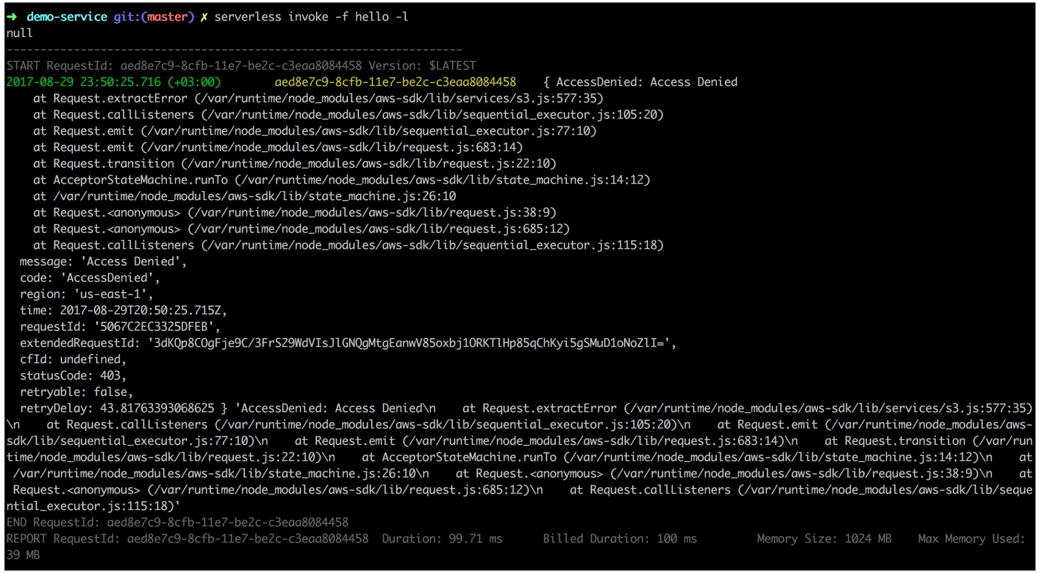
```
'use strict';
var AWS = require('aws-sdk');
var s3 = new AWS.S3();

module.exports.hello = (event, context, callback) => {
  var params = {
    Bucket: 'serverless-tutorial-techgeeks-12345',
  };

  s3.listObjectsV2(params, function(err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      const response = {
        statusCode: 200,
        body: JSON.stringify({
          "bucket_list": data
        }),
      };
      callback(null, response);
    }
  });
};
```

Read files from S3

```
$ serverless invoke -f hello -l
```

You should get an ‘Access denied’ error.

Your service doesn’t have permissions to read from S3 buckets



```
➜ demo-service git:(master) ✗ serverless invoke -f hello -l
null
--------------------------------------------------------------------
START RequestId: aed8e7c9-8cfb-11e7-be2c-c3eaa8084458 Version: $LATEST
2017-08-29 23:50:25.716 (+03:00)        aed8e7c9-8cfb-11e7-be2c-c3eaa8084458    { AccessDenied: Access Denied
    at Request.extractError (/var/runtime/node_modules/aws-sdk/lib/services/s3.js:577:35)
    at Request.callListeners (/var/runtime/node_modules/aws-sdk/lib/sequential_executor.js:105:20)
    at Request.emit (/var/runtime/node_modules/aws-sdk/lib/sequential_executor.js:77:10)
    at Request.emit (/var/runtime/node_modules/aws-sdk/lib/request.js:683:14)
    at Request.transition (/var/runtime/node_modules/aws-sdk/lib/request.js:22:10)
    at AcceptorStateMachine.runTo (/var/runtime/node_modules/aws-sdk/lib/state_machine.js:14:12)
    at /var/runtime/node_modules/aws-sdk/lib/state_machine.js:26:10
    at Request.<anonymous> (/var/runtime/node_modules/aws-sdk/lib/request.js:38:9)
    at Request.<anonymous> (/var/runtime/node_modules/aws-sdk/lib/request.js:685:12)
    at Request.callListeners (/var/runtime/node_modules/aws-sdk/lib/sequential_executor.js:115:18)
  message: 'Access Denied',
  code: 'AccessDenied',
  region: 'us-east-1',
  time: 2017-08-29T20:50:25.715Z,
  requestId: '5067C2EC3325DFEB',
  extendedRequestId: '3dKQp8COgFje9C/3FrS29WdVIsJlGNQgMtgEanwV85oxbj1ORKTlHp85qChKyi5gSMuD1oNoZlI=',
  cfId: undefined,
  statusCode: 403,
  retryable: false,
  retryDelay: 43.81763393068625 } 'AccessDenied: Access Denied\n    at Request.extractError (/var/runtime/node_modules/aws-sdk/lib/services/s3.js:577:35)
\n    at Request.callListeners (/var/runtime/node_modules/aws-sdk/lib/sequential_executor.js:105:20)\n    at Request.emit (/var/runtime/node_modules/aws-
sdk/lib/sequential_executor.js:77:10)\n    at Request.emit (/var/runtime/node_modules/aws-sdk/lib/request.js:683:14)\n    at Request.transition (/var/run
time/node_modules/aws-sdk/lib/request.js:22:10)\n    at AcceptorStateMachine.runTo (/var/runtime/node_modules/aws-sdk/lib/state_machine.js:14:12)\n    at
 /var/runtime/node_modules/aws-sdk/lib/state_machine.js:26:10\n    at Request.<anonymous> (/var/runtime/node_modules/aws-sdk/lib/request.js:38:9)\n    at
 Request.<anonymous> (/var/runtime/node_modules/aws-sdk/lib/request.js:685:12)\n    at Request.callListeners (/var/runtime/node_modules/aws-sdk/lib/seque
ntial_executor.js:115:18)'
END RequestId: aed8e7c9-8cfb-11e7-be2c-c3eaa8084458
REPORT RequestId: aed8e7c9-8cfb-11e7-be2c-c3eaa8084458  Duration: 99.71 ms      Billed Duration: 100 ms         Memory Size: 1024 MB    Max Memory Used:
39 MB
```

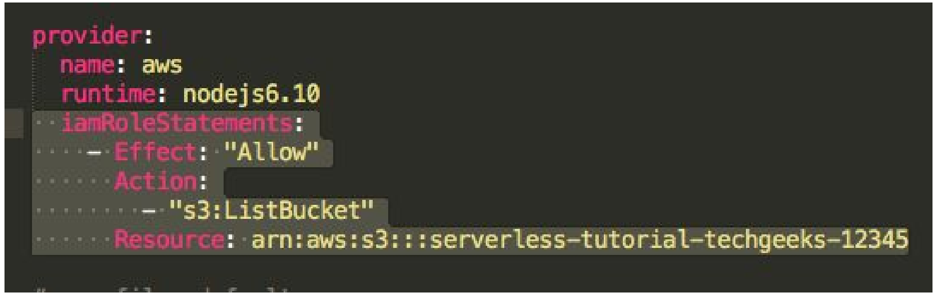
Read files from S3

Assign an IAM Role with S3 *ListBucket* permissions in serverless.yml

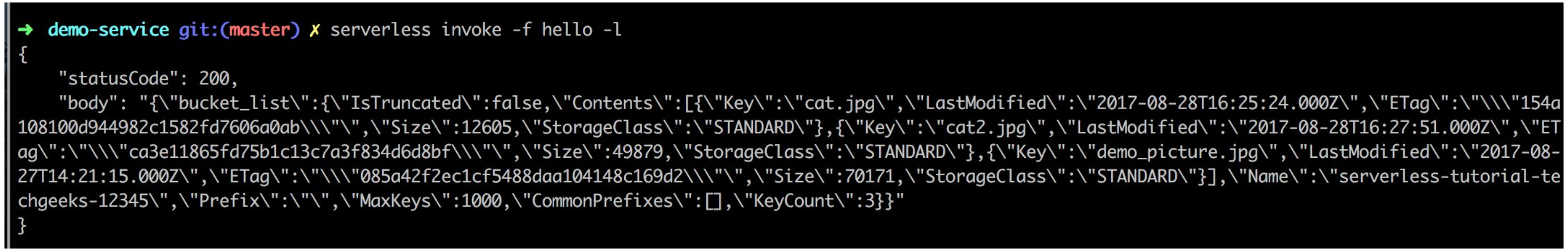
* replace bucket name in Resource field with yours

Invoke function again:

```
$ serverless invoke -f hello -l
```



```
provider:
  name: aws
  runtime: nodejs6.10
  iamRoleStatements:
    - Effect: "Allow"
      Action:
        - "s3:ListBucket"
      Resource: arn:aws:s3:::serverless-tutorial-techgeeks-12345
```

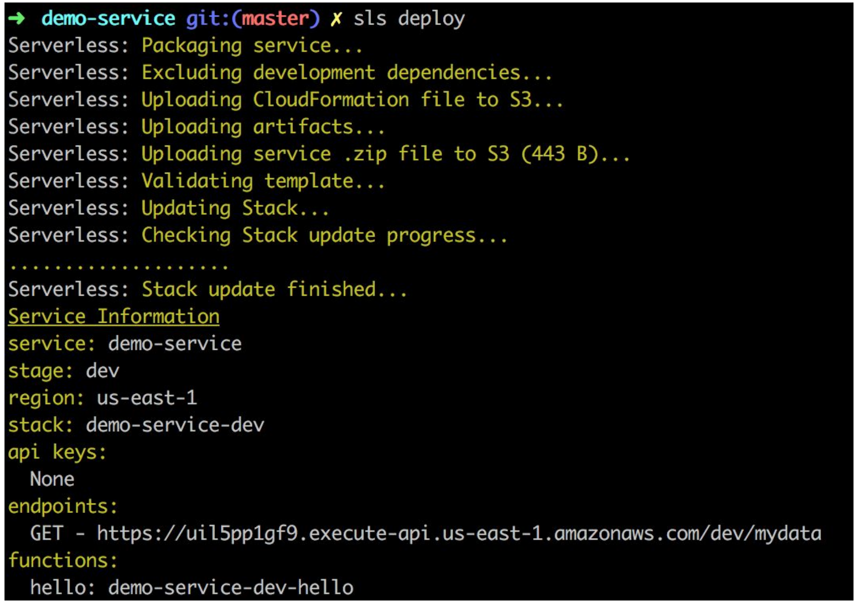


```
➜  demo-service git:(master) ✗ serverless invoke -f hello -l
{
    "statusCode": 200,
    "body": "{\"bucket_list\":{\"IsTruncated\":false,\"Contents\":[{\"Key\":\"cat.jpg\",\"LastModified\":\"2017-08-28T16:25:24.000Z\",\"ETag\":\"\\\"154a
108100d944982c1582fd7606a0ab\\\"\",\"Size\":12605,\"StorageClass\":\"STANDARD\"},{\"Key\":\"cat2.jpg\",\"LastModified\":\"2017-08-28T16:27:51.000Z\",\"ET
ag\":\"\\\"ca3e11865fd75b1c13c7a3f834d6d8bf\\\"\",\"Size\":49879,\"StorageClass\":\"STANDARD\"},{\"Key\":\"demo_picture.jpg\",\"LastModified\":\"2017-08-
27T14:21:15.000Z\",\"ETag\":\"\\\"085a42f2ec1cf5488daa104148c169d2\\\"\",\"Size\":70171,\"StorageClass\":\"STANDARD\"}],\"Name\":\"serverless-tutorial-te
chgeeks-12345\",\"Prefix\":\"\",\"MaxKeys\":1000,\"CommonPrefixes\":[],\"KeyCount\":3}}"
}
```

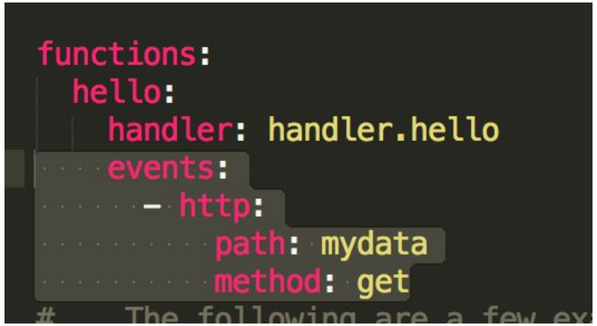
Endpoint

Add endpoint in serverless.yml

```
$ serverless deploy
```



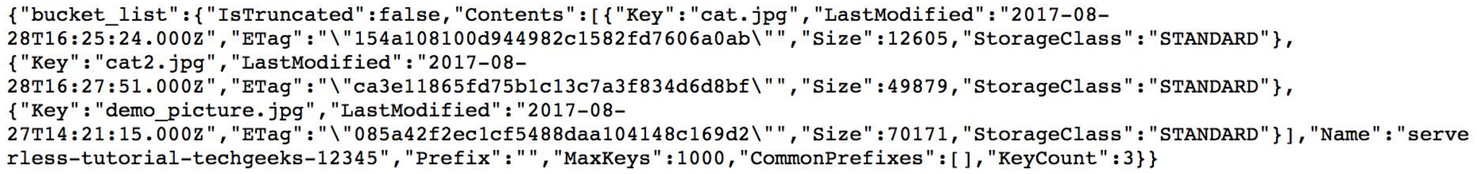
```
➜ demo-service git:(master) ✗ sls deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service .zip file to S3 (443 B)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
....................
Serverless: Stack update finished...
Service Information
service: demo-service
stage: dev
region: us-east-1
stack: demo-service-dev
api keys:
  None
endpoints:
  GET - https://uil5pp1gf9.execute-api.us-east-1.amazonaws.com/dev/mydata
functions:
  hello: demo-service-dev-hello
```



```
functions:
  hello:
    handler: handler.hello
    events:
      - http:
          path: mydata
          method: get
#     The following are a few ex
```

Endpoint

Access GET endpoint reported in console after deploy and you should see something like:



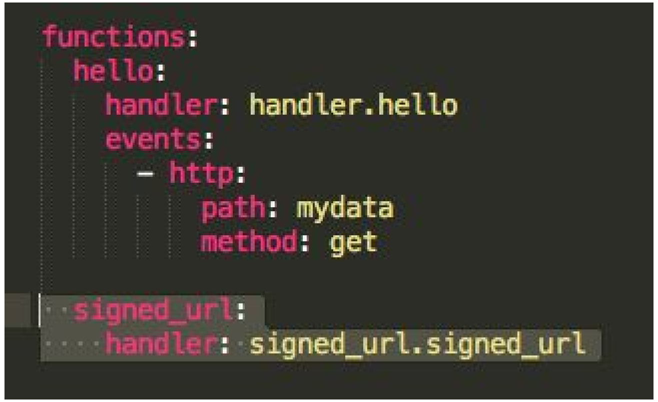
```
{"bucket_list":{"IsTruncated":false,"Contents":[{"Key":"cat.jpg","LastModified":"2017-08-
28T16:25:24.000Z","ETag":"\"154a108100d944982c1582fd7606a0ab\"","Size":12605,"StorageClass":"STANDARD"},
{"Key":"cat2.jpg","LastModified":"2017-08-
28T16:27:51.000Z","ETag":"\"ca3e11865fd75b1c13c7a3f834d6d8bf\"","Size":49879,"StorageClass":"STANDARD"},
{"Key":"demo_picture.jpg","LastModified":"2017-08-
27T14:21:15.000Z","ETag":"\"085a42f2ec1cf5488daa104148c169d2\"","Size":70171,"StorageClass":"STANDARD"}],"Name":"serve
rless-tutorial-techgeeks-12345","Prefix":"","MaxKeys":1000,"CommonPrefixes":[],"KeyCount":3}}
```

3. Events

Create a new function *signed_url* that runs when an object is uploaded to an S3 bucket and logs a signed url for that item.

Events

Define new function in serverless.yml



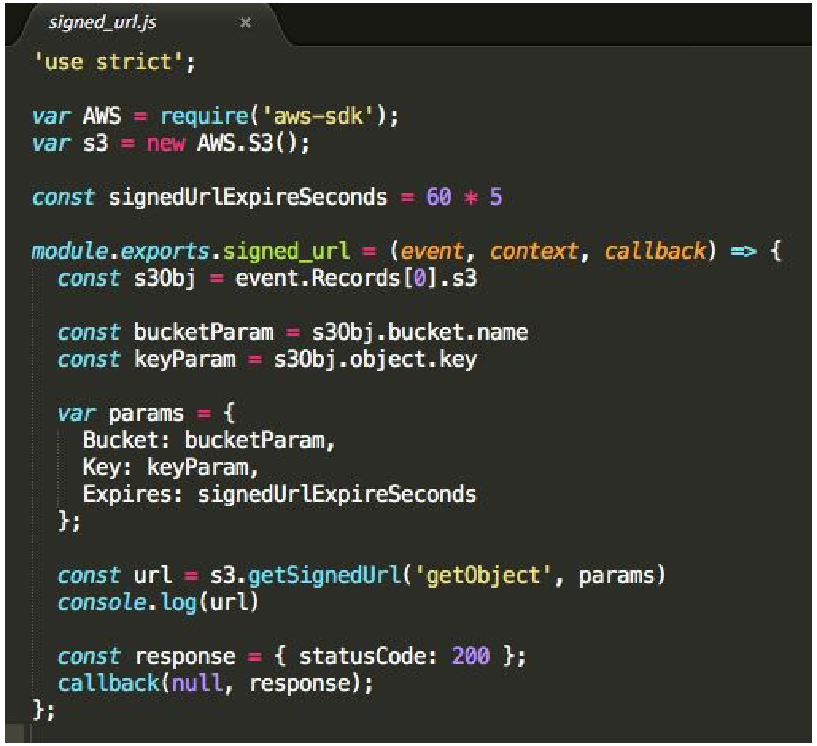
```yaml
functions:
  hello:
    handler: handler.hello
    events:
      - http:
          path: mydata
          method: get

  signed_url:
    handler: signed_url.signed_url
```

Events

Create signed_url.js with the code to the right.

Deploy service (anytime you change serverless.yml you need to deploy the whole service)

```
$ serverless deploy
```

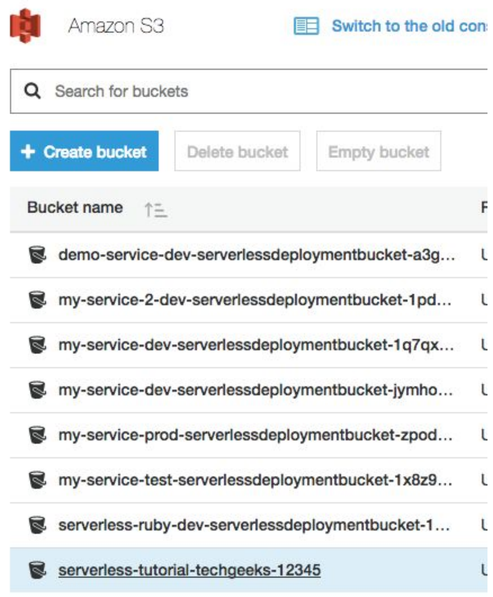


signed_url.js

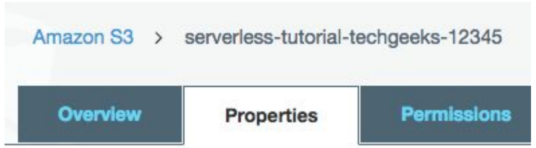
```
'use strict';

var AWS = require('aws-sdk');
var s3 = new AWS.S3();

const signedUrlExpireSeconds = 60 * 5

module.exports.signed_url = (event, context, callback) => {
  const s3Obj = event.Records[0].s3

  const bucketParam = s3Obj.bucket.name
  const keyParam = s3Obj.object.key

  var params = {
    Bucket: bucketParam,
    Key: keyParam,
    Expires: signedUrlExpireSeconds
  };

  const url = s3.getSignedUrl('getObject', params)
  console.log(url)

  const response = { statusCode: 200 };
  callback(null, response);
};
```

Events

Go to your AWS S3 bucket -> Properties -> Events

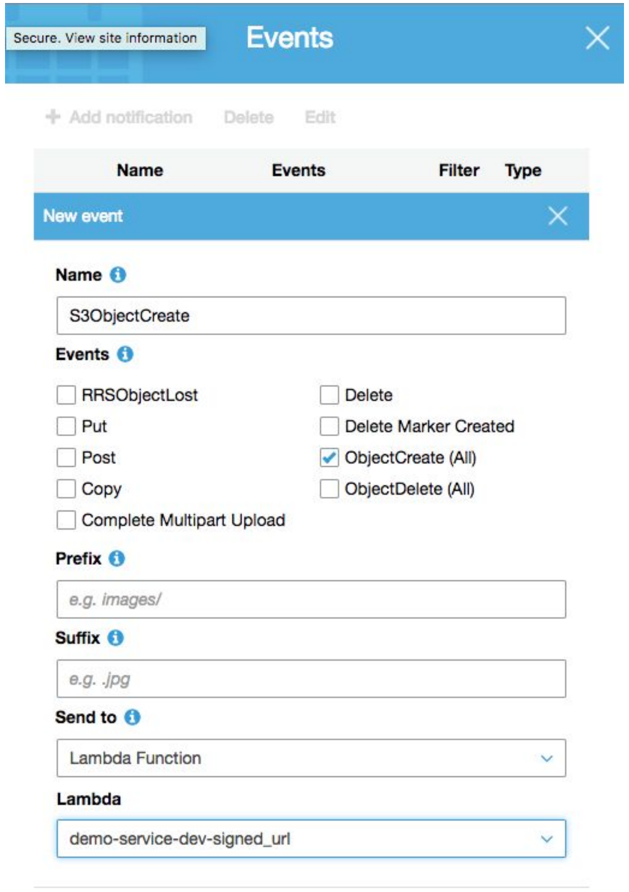


Bucket name
demo-service-dev-serverlessdeploymentbucket-a3g...
my-service-2-dev-serverlessdeploymentbucket-1pd...
my-service-dev-serverlessdeploymentbucket-1q7qx...
my-service-dev-serverlessdeploymentbucket-jymho...
my-service-prod-serverlessdeploymentbucket-zpod...
my-service-test-serverlessdeploymentbucket-1x8z9...
serverless-ruby-dev-serverlessdeploymentbucket-1...
serverless-tutorial-techgeeks-12345



Events

Create Notification for *ObjectCreate (All)* event, and set it to notify your lambda function.



Name: S3ObjectCreate

Events:

- [ ] RRSObjectLost
- [ ] Put
- [ ] Post
- [ ] Copy
- [ ] Complete Multipart Upload
- [ ] Delete
- [ ] Delete Marker Created
- [✓] ObjectCreate (All)
- [ ] ObjectDelete (All)

Prefix: *e.g. images/*

Suffix: *e.g. .jpg*

Send to: Lambda Function

Lambda: demo-service-dev-signed_url

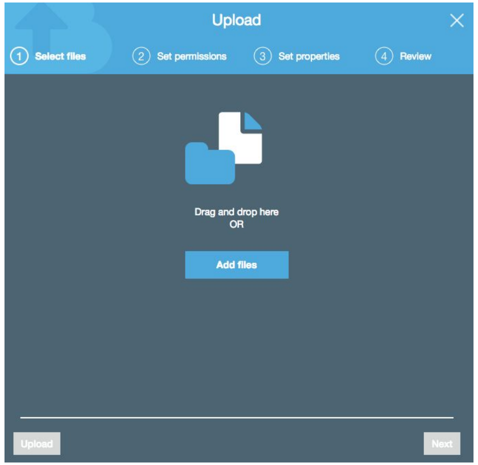
Events

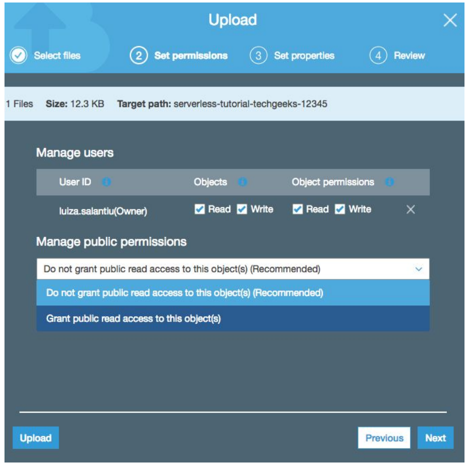
Open a new console and stream logs for the function. When you upload an image to your bucket, you should see the function being call in the logs.

```
$ serverless logs -f signed_url -t
```

Upload image to your S3 bucket

* grant public access for now; normally it wouldn't be needed - signed urls are for providing short lived access to an object that is not publicly available but for some reason when the function is ran in AWS, it gives a weird signed url. Different than if code runs locally)





1 Files **Size:** 12.3 KB **Target path:** serverless-tutorial-techgeeks-12345

Manage users

User ID	Objects	Object permissions
luiza.salantiu(Owner)	✓ Read ✓ Write	✓ Read ✓ Write

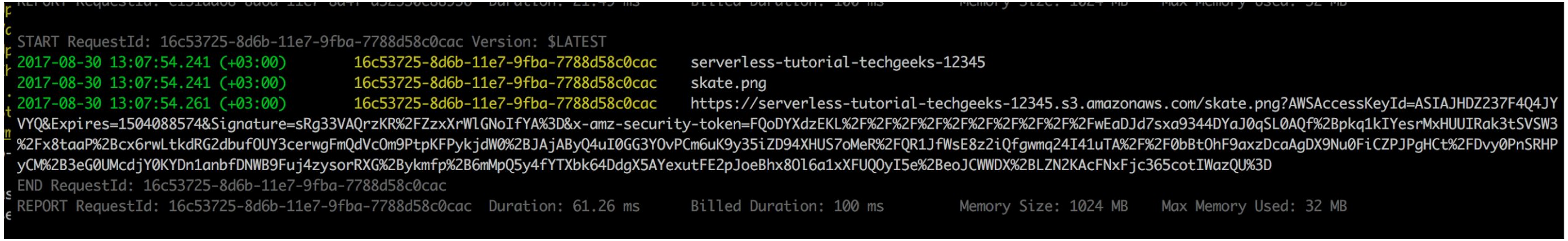
Manage public permissions

Do not grant public read access to this object(s) (Recommended)

- Do not grant public read access to this object(s) (Recommended)
- Grant public read access to this object(s)

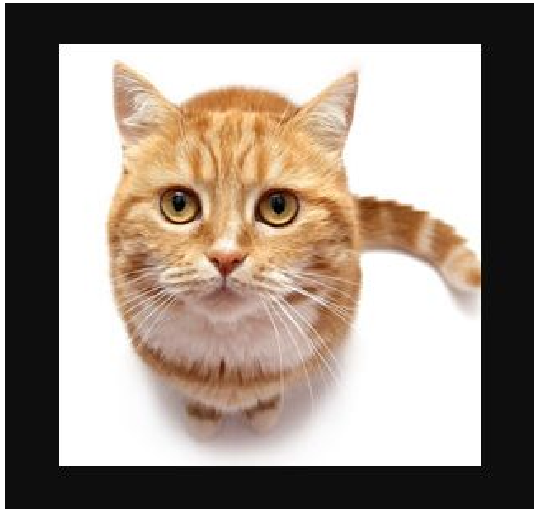
Wait for it ...

Events



```
START RequestId: 16c53725-8d6b-11e7-9fba-7788d58c0cac Version: $LATEST
2017-08-30 13:07:54.241 (+03:00)        16c53725-8d6b-11e7-9fba-7788d58c0cac    serverless-tutorial-techgeeks-12345
2017-08-30 13:07:54.241 (+03:00)        16c53725-8d6b-11e7-9fba-7788d58c0cac    skate.png
2017-08-30 13:07:54.261 (+03:00)        16c53725-8d6b-11e7-9fba-7788d58c0cac    https://serverless-tutorial-techgeeks-12345.s3.amazonaws.com/skate.png?AWSAccessKeyId=ASIAJHDZ237F4Q4JY
VYQ&Expires=1504088574&Signature=sRg33VAQrzKR%2FZzxXrWlGNoIfYA%3D&x-amz-security-token=FQoDYXdzEKL%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaDJd7sxa9344DYaJ0qSL0AQf%2Bpkq1kIYesrMxHUUIRak3tSVSW3
%2Fx8taaP%2Bcx6rwLtkdRG2dbufOUY3cerwgFmQdVcOm9PtpKFPykjdW0%2BJAjAByQ4uI0GG3YOvPCm6uK9y35iZD94XHUS7oMeR%2FQR1JfWsE8z2iQfgwmq24I41uTA%2F%2F0bBtOhF9axzDcaAgDX9Nu0FiCZPJPgHCt%2FDvy0PnSRHP
yCM%2B3eG0UMcdjY0KYDn1anbfDNWB9Fuj4zysorRXG%2Bykmfp%2B6mMpQ5y4fYTXbk64DdgX5AYexutFE2pJoeBhx8Ol6a1xXFUQOyI5e%2BeoJCWWDX%2BLZN2KAcFNxFjc365cotIWazQU%3D
END RequestId: 16c53725-8d6b-11e7-9fba-7788d58c0cac
REPORT RequestId: 16c53725-8d6b-11e7-9fba-7788d58c0cac  Duration: 61.26 ms      Billed Duration: 100 ms         Memory Size: 1024 MB    Max Memory Used: 32 MB
```

Voila! If you open the link in your browser you should see your picture



Resources

https://martinfowler.com/articles/serverless.html
https://github.com/anaibol/awesome-serverless
https://aws.amazon.com/lambda/
http://docs.aws.amazon.com/cli/latest/reference/
http://docs.aws.amazon.com/cli/latest/reference/lambda/list-functions.html
https://github.com/awslabs?utf8=%E2%9C%93&q=serverless&type=&language=

https://github.com/serverless/guide
https://serverless.com/framework/docs/providers/aws/cli-reference/
https://github.com/serverless/examples
https://github.com/serverless/dashboard

Tutorials
http://serverless-stack.com/
Building a REST API