AWS Lambda and API Gateway

Getting started with serverless REST APIs

Karol Horosin

September, 2nd 2020

process

ardigen

Artificial Intelligence & Bioinformatics
for Precision Medicine

CODE AGAINST CANCER

karol.horosin@gmail.com

Presentation plan

We’re going to discuss:

1. Basic serverless concepts
2. What are we going to build?
3. Manual setup
4. Serverless framework
5. Let’s code an API
6. Gitlab CI/CD setup
7. Recommended resources

serverless computing

a method of providing services on an **as-used basis**. Serverless architecture allows users to **write and deploy code without the hassle of worrying about the underlying infrastructure**

FaaS

Function as a service

a serverless way to execute modular pieces of code. FaaS lets developers write and update a piece of code on the fly, which can then be **executed in response to an event**, such as a user clicking on an element in a web application, object being added to the database, etc.

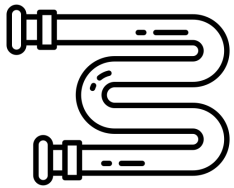
What are we going to build?

Basic rest API with simple endpoints you can build upon.

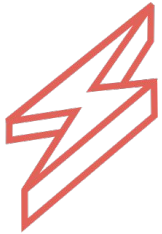
(including configured development process)

CI/CD

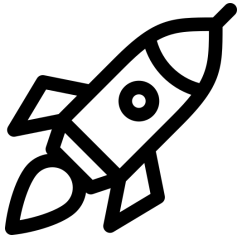
1. Push to repo



2. Trigger CI/CD pipeline



3. Install sls in node container



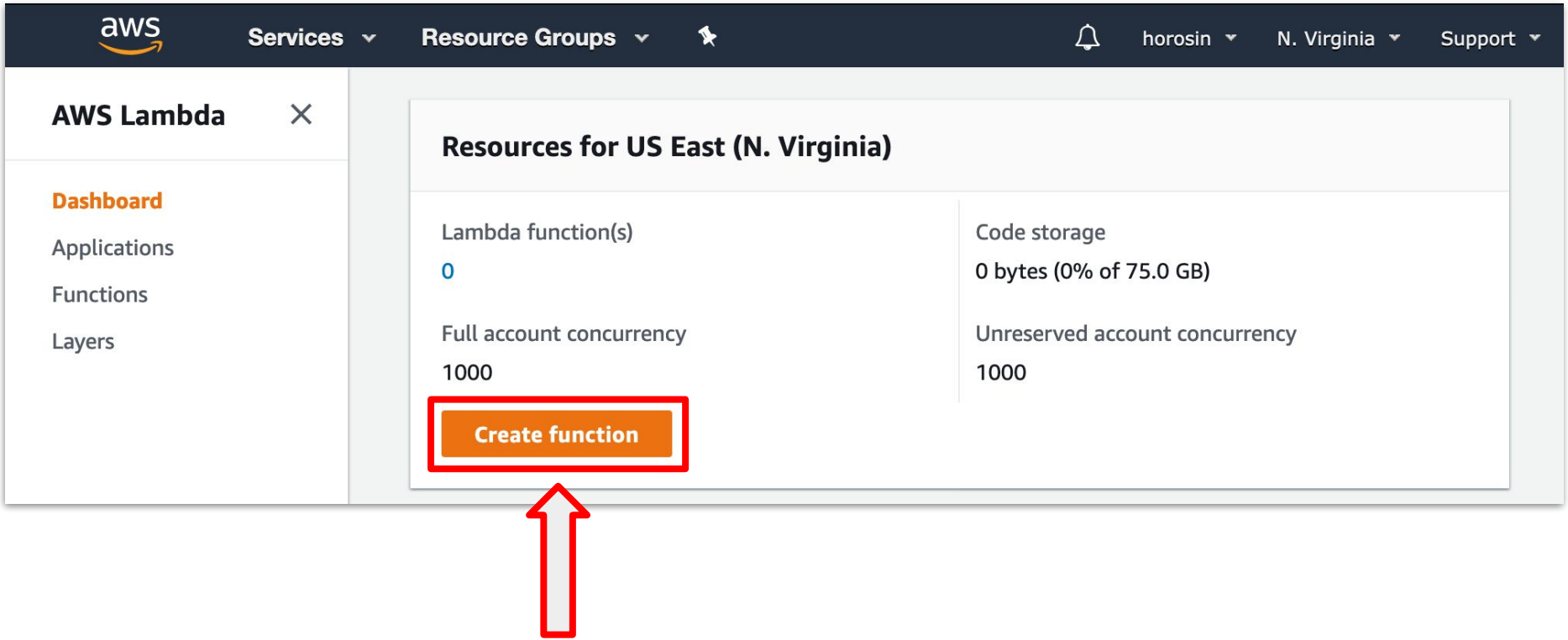
4. Deploy!

Manual lambda setup

Manual lambda setup (not great)

Go to: https://eu-central-1.console.aws.amazon.com/lambda/home

And choose “Create function” or
“Functions” from the menu and then “Create function”.


aws
Services
Resource Groups
horosin
N. Virginia
Support
AWS Lambda
Dashboard
Applications
Functions
Layers
Resources for US East (N. Virginia)
Lambda function(s)
0
Code storage
0 bytes (0% of 75.0 GB)
Full account concurrency
1000
Unreserved account concurrency
1000
Create function

Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

Author from scratch

Start with a simple Hello World example.

Use a blueprint

Build a Lambda application from sample code and configuration presets for common use cases.

Browse serverless app repository

Deploy a sample Lambda application from the AWS Serverless Application Repository.

presets for common use cases.

from the AWS Serverless Application Repository.

Blueprints Info

Export

Add filter

1

Keyword : hello-world

hello-world

A starter AWS Lambda function.

nodejs

hello-world-python

A starter AWS Lambda function.

python3.7

Cancel

Configure

Basic information Info

Function name

basic-function

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the **IAM console**.

- (•) Create a new role with basic Lambda permissions
- ( ) Use an existing role
- ( ) Create a new role from AWS policy templates

ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named basic-function-role-fy8ij4wx, with permission to upload logs to Amazon CloudWatch Logs.

Lambda function code

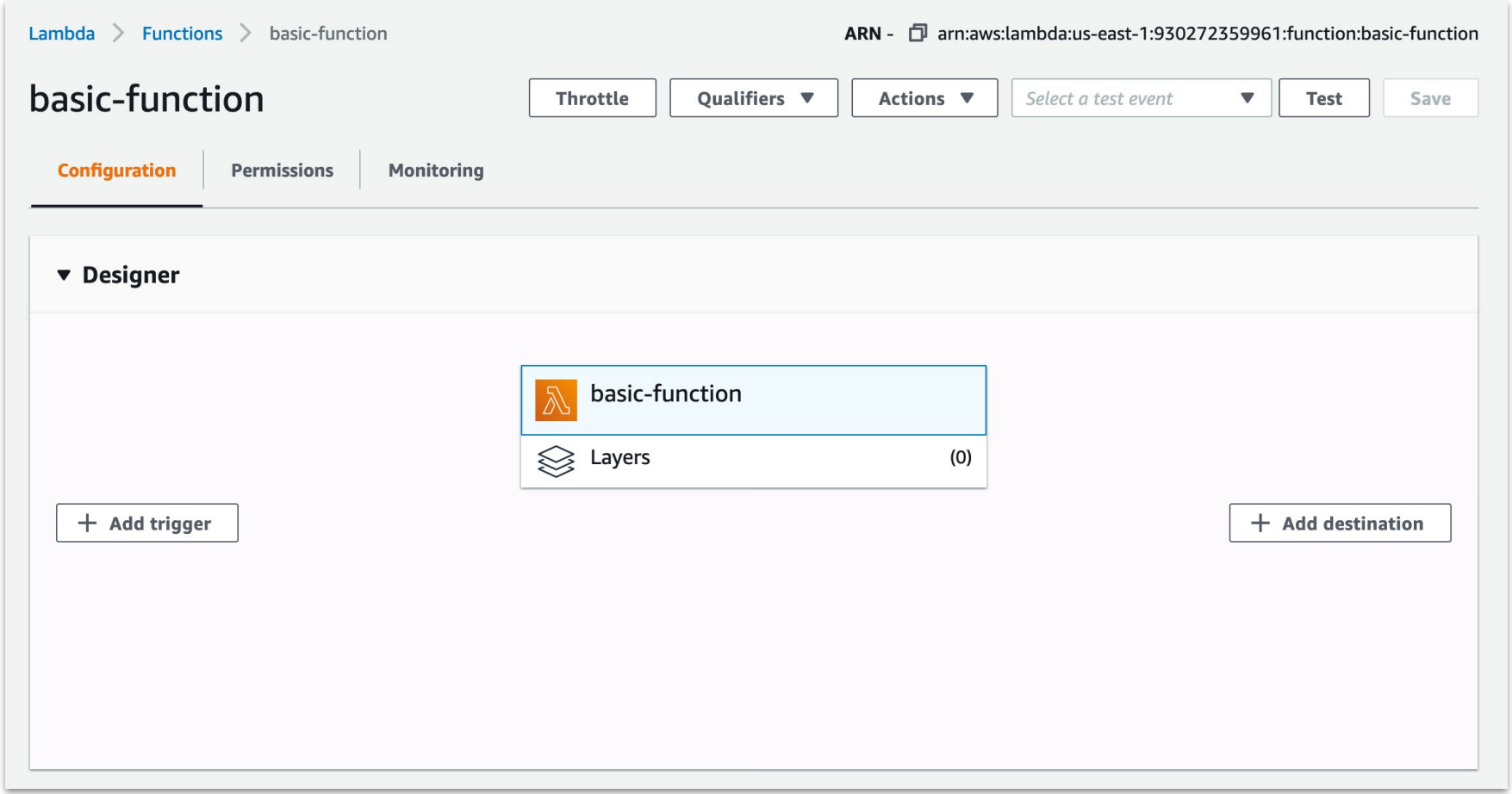
Code is preconfigured by the chosen blueprint. You can configure it after you create the function. **Learn more** about deploying Lambda functions.

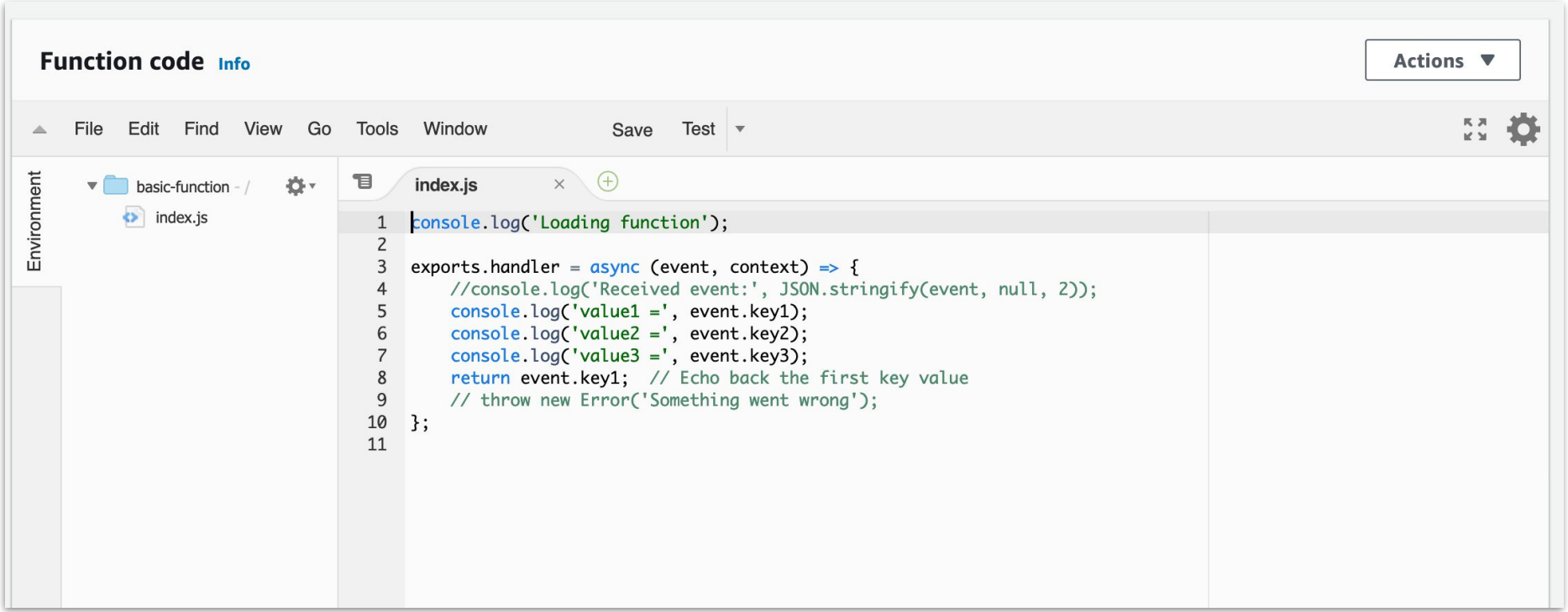
Runtime

Node.js 12.x

```
1  console.log('Loading function');
2
3  exports.handler = async (event, context) => {
4      //console.log('Received event:', JSON.stringify(event, null, 2));
5      console.log('value1 =', event.key1);
6      console.log('value2 =', event.key2);
7      console.log('value3 =', event.key3);
8      return event.key1;  // Echo back the first key value
9      // throw new Error('Something went wrong');
10 };
11
```

Cancel | Create function


Lambda
Functions
basic-function
ARN - arn:aws:lambda:us-east-1:930272359961:function:basic-function
basic-function
Throttle
Qualifiers
Actions
Select a test event
Test
Save
Configuration
Permissions
Monitoring
Designer
basic-function
Layers
(0)
Add trigger
Add destination


Function code Info
Actions
File Edit Find View Go Tools Window Save Test
Environment
basic-function - /
index.js
index.js
1 console.log('Loading function');
2
3 exports.handler = async (event, context) => {
4 //console.log('Received event:', JSON.stringify(event, null, 2));
5 console.log('value1 =', event.key1);
6 console.log('value2 =', event.key2);
7 console.log('value3 =', event.key3);
8 return event.key1; // Echo back the first key value
9 // throw new Error('Something went wrong');
10 };
11

Environment variables (0)

Edit

Key	Value

No environment variables

No environment variables associated with this function.

Manage environment variables

Tags (1)

Manage tags

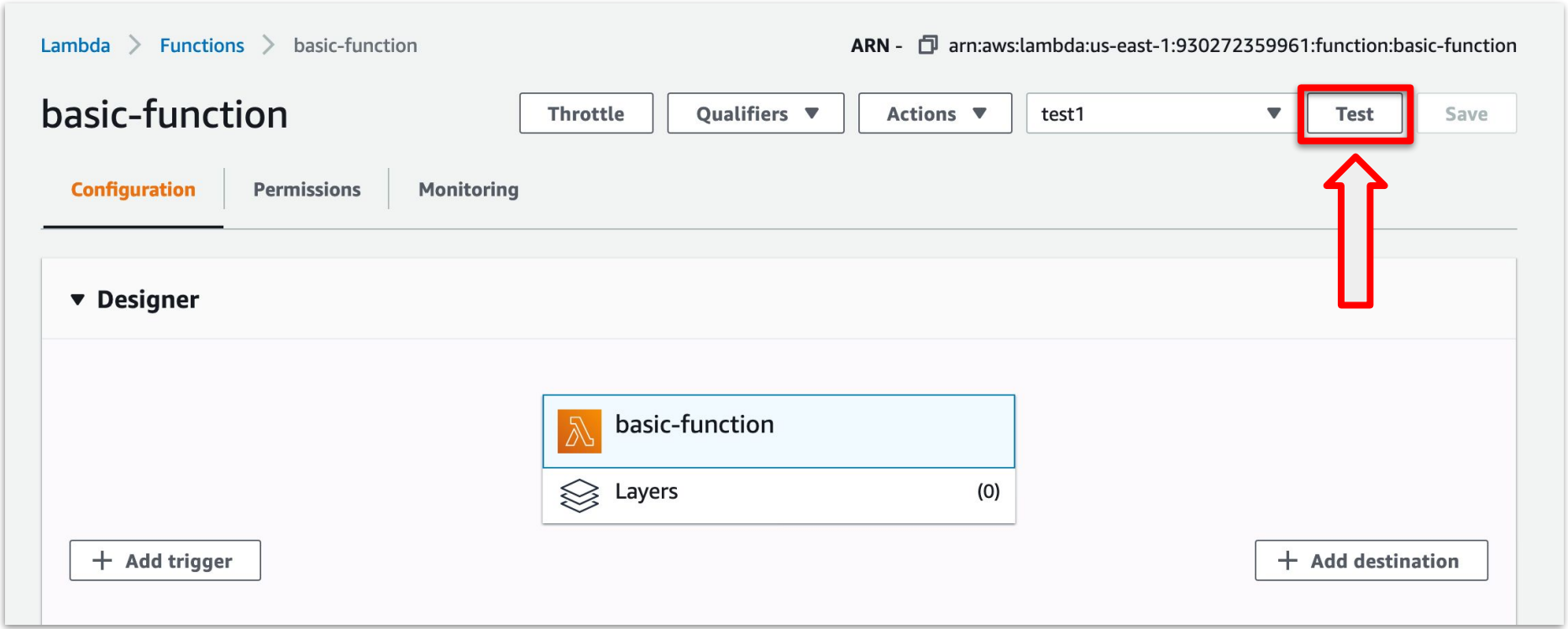
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

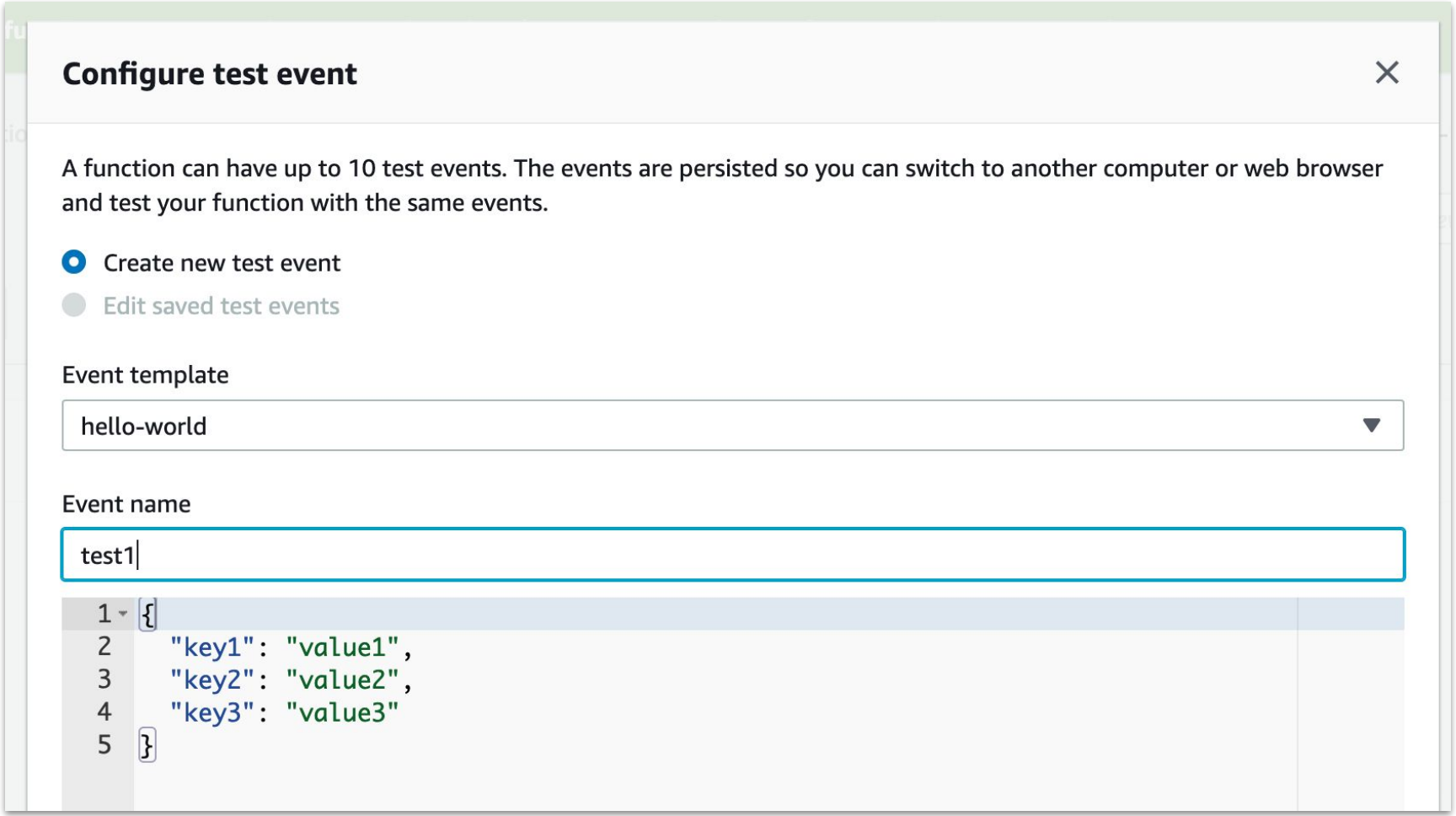
Key	Value
lambda-console:blueprint	hello-world

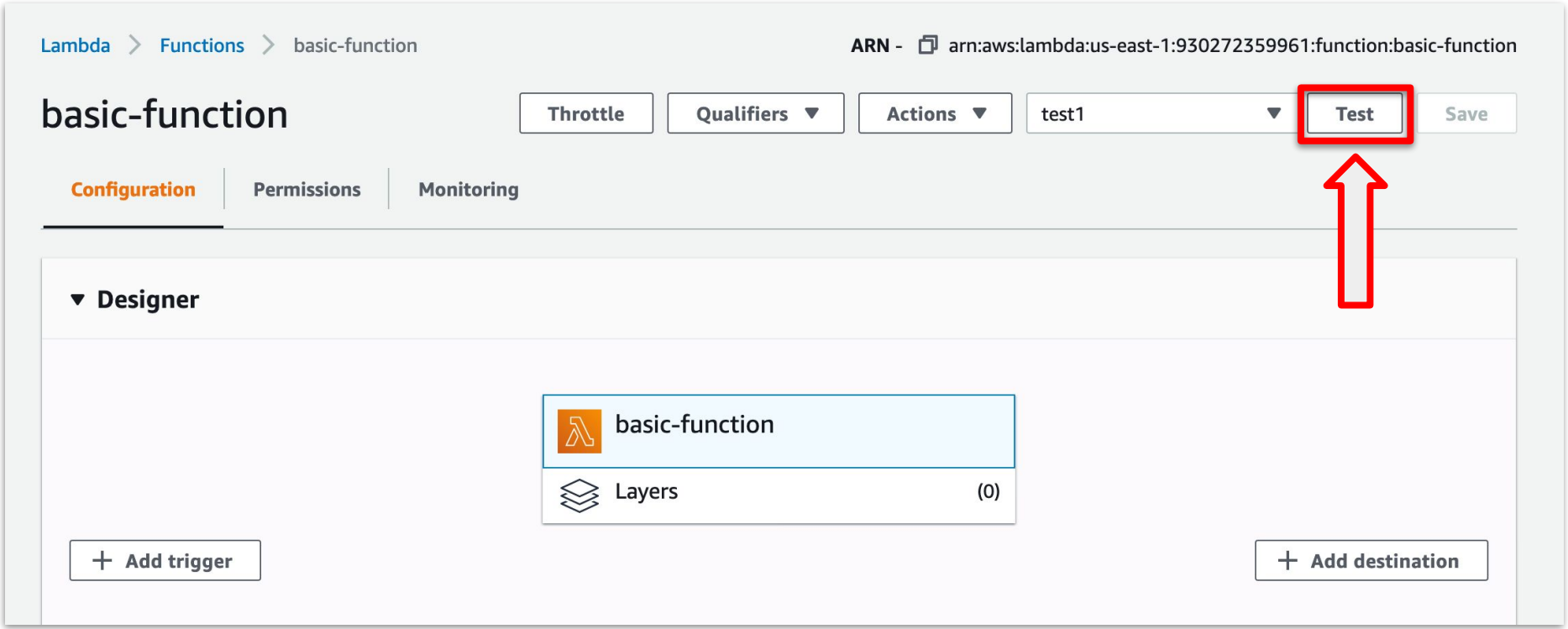
Basic settings

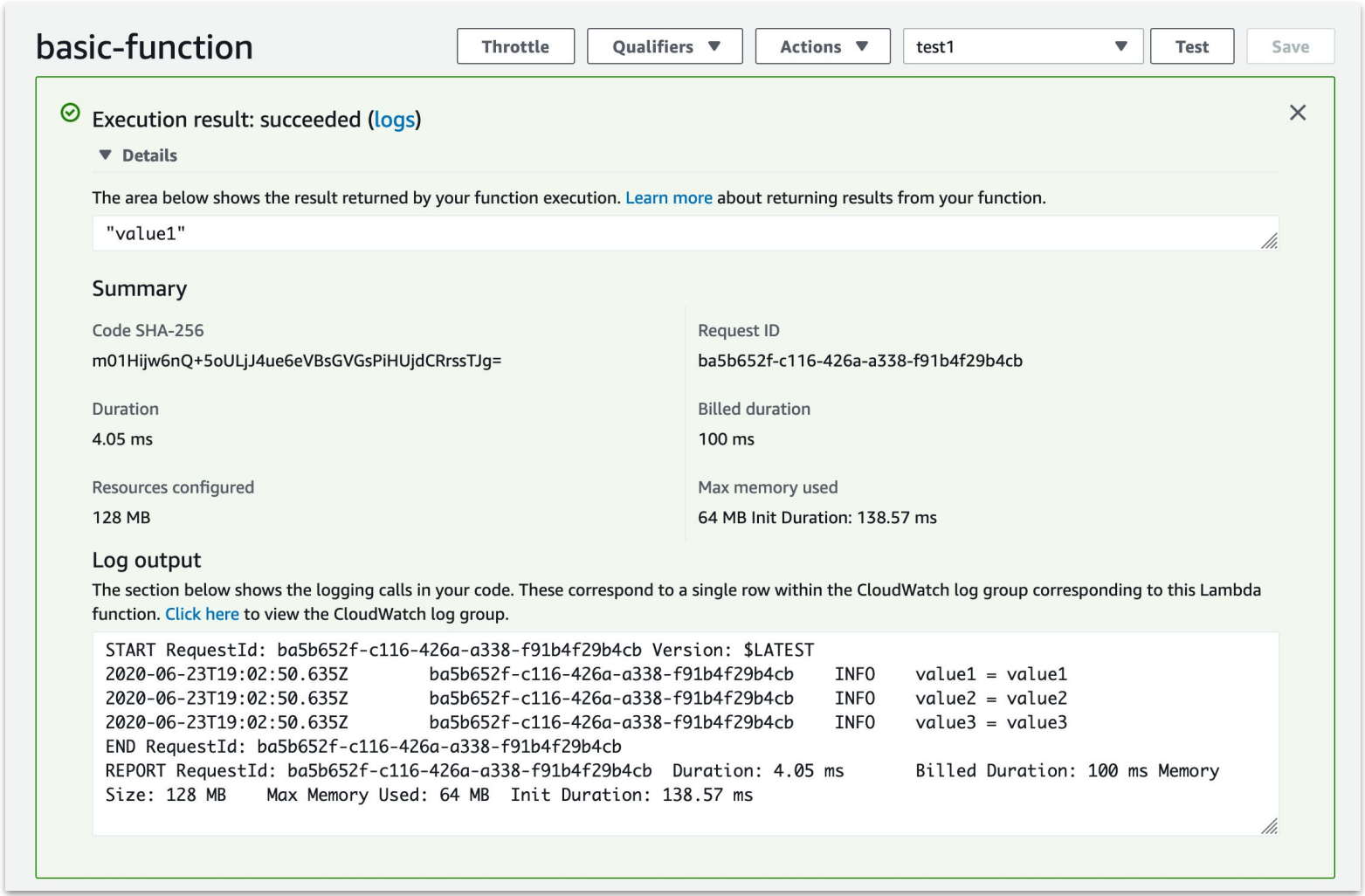
Edit

AWS X-Ray Info

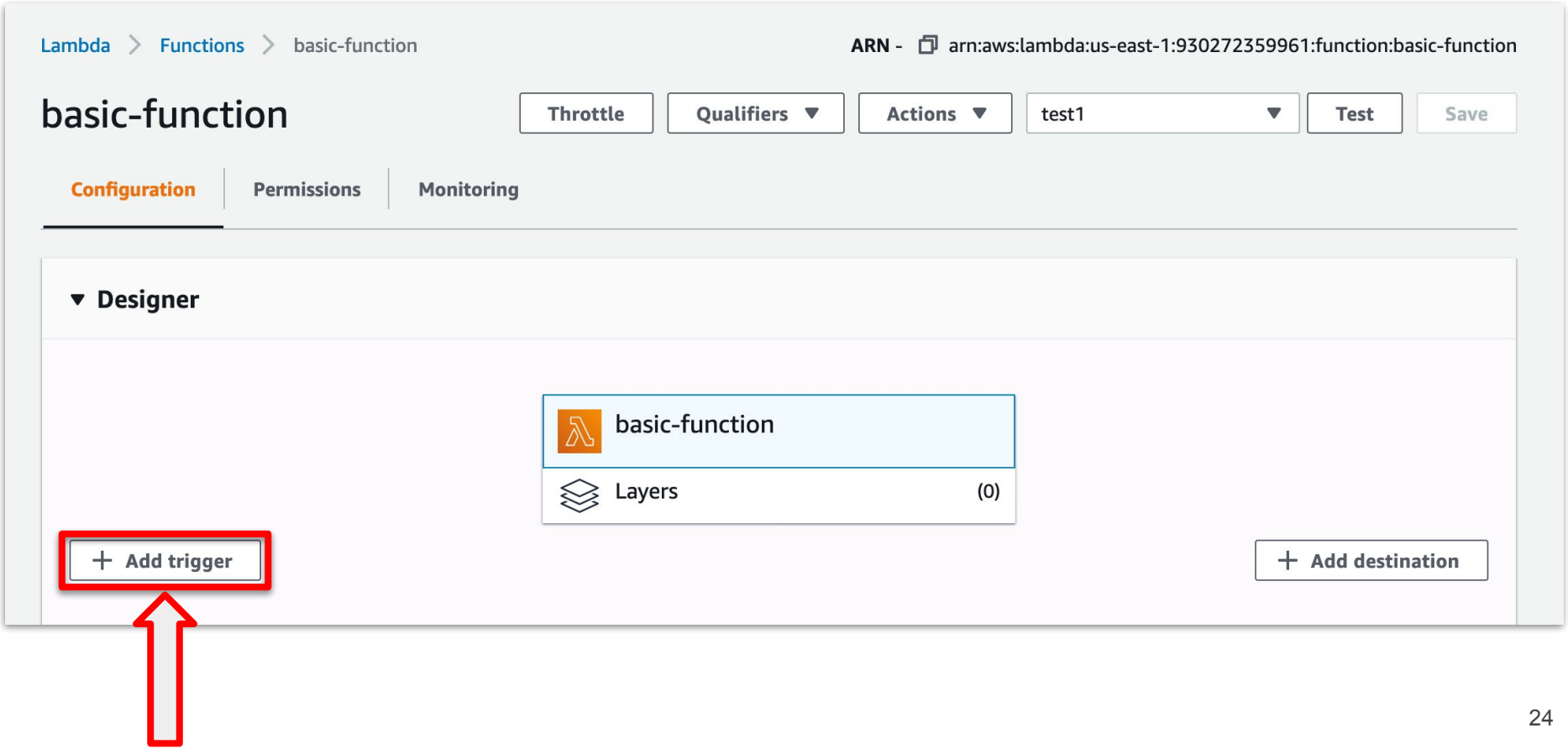

Lambda
Functions
basic-function
ARN - arn:aws:lambda:us-east-1:930272359961:function:basic-function
basic-function
Throttle
Qualifiers
Actions
test1
Test
Save
Configuration
Permissions
Monitoring
Designer
basic-function
Layers
(0)
Add trigger
Add destination

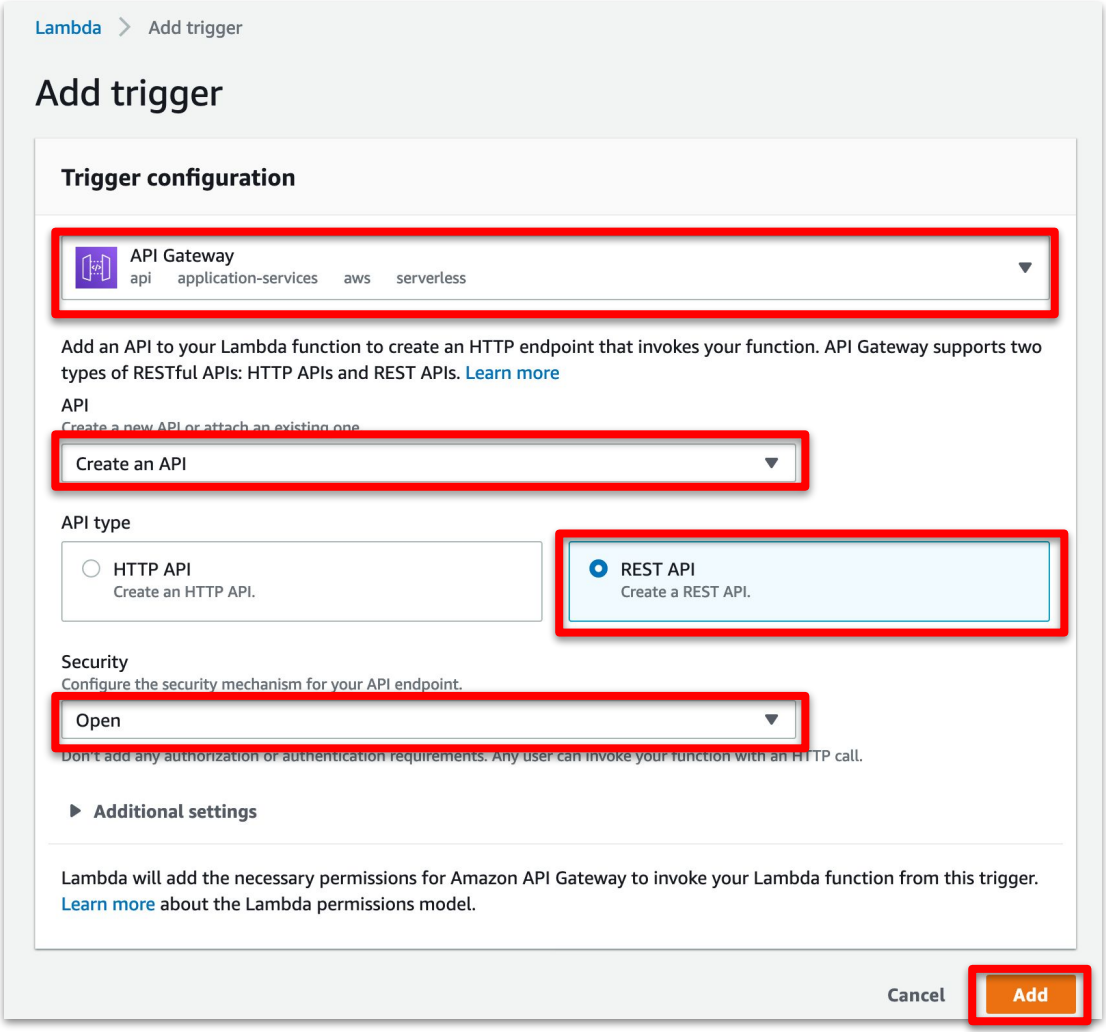

Configure test event
A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.
Create new test event
Edit saved test events
Event template
hello-world
Event name
test1
{
"key1": "value1",
"key2": "value2",
"key3": "value3"
}

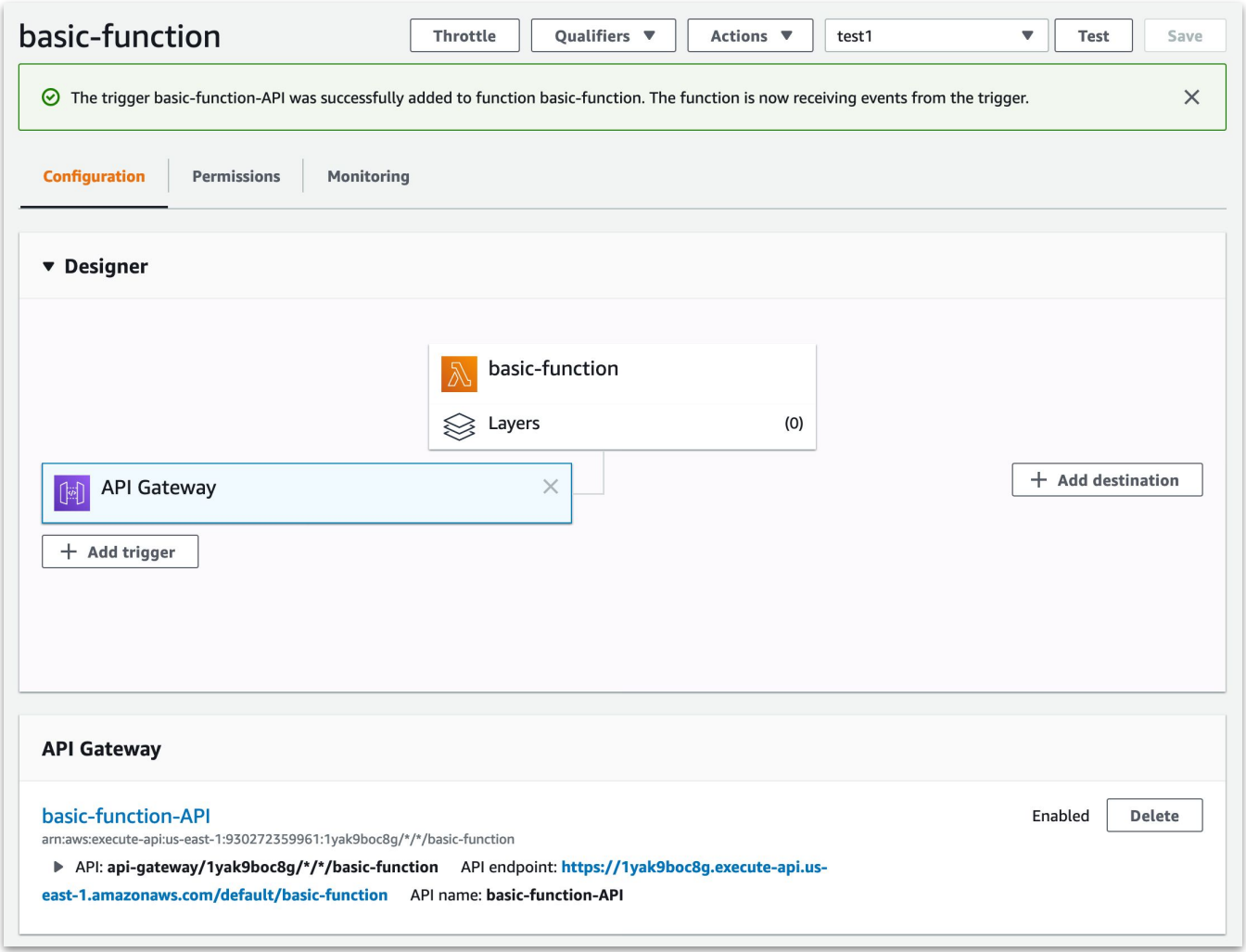

Lambda
Functions
basic-function
ARN - arn:aws:lambda:us-east-1:930272359961:function:basic-function
basic-function
Throttle
Qualifiers
Actions
test1
Test
Save
Configuration
Permissions
Monitoring
Designer
basic-function
Layers
(0)
Add trigger
Add destination

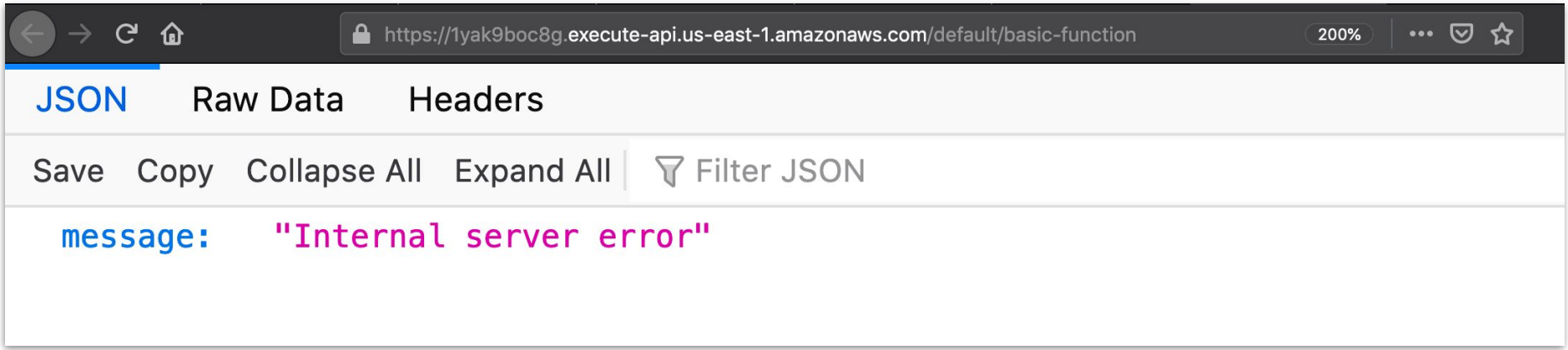

basic-function
Throttle
Qualifiers
Actions
test1
Test
Save
Execution result: succeeded (logs)
Details
The area below shows the result returned by your function execution. Learn more about returning results from your function.
"value1"
Summary
Code SHA-256
m01Hijw6nQ+5oULjJ4ue6eVBsGVGsPiHUjdCRrssTJg=
Request ID
ba5b652f-c116-426a-a338-f91b4f29b4cb
Duration
4.05 ms
Billed duration
100 ms
Resources configured
128 MB
Max memory used
64 MB Init Duration: 138.57 ms
Log output
The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.
START RequestId: ba5b652f-c116-426a-a338-f91b4f29b4cb Version: $LATEST
2020-06-23T19:02:50.635Z ba5b652f-c116-426a-a338-f91b4f29b4cb INFO value1 = value1
2020-06-23T19:02:50.635Z ba5b652f-c116-426a-a338-f91b4f29b4cb INFO value2 = value2
2020-06-23T19:02:50.635Z ba5b652f-c116-426a-a338-f91b4f29b4cb INFO value3 = value3
END RequestId: ba5b652f-c116-426a-a338-f91b4f29b4cb
REPORT RequestId: ba5b652f-c116-426a-a338-f91b4f29b4cb Duration: 4.05 ms Billed Duration: 100 ms Memory
Size: 128 MB Max Memory Used: 64 MB Init Duration: 138.57 ms

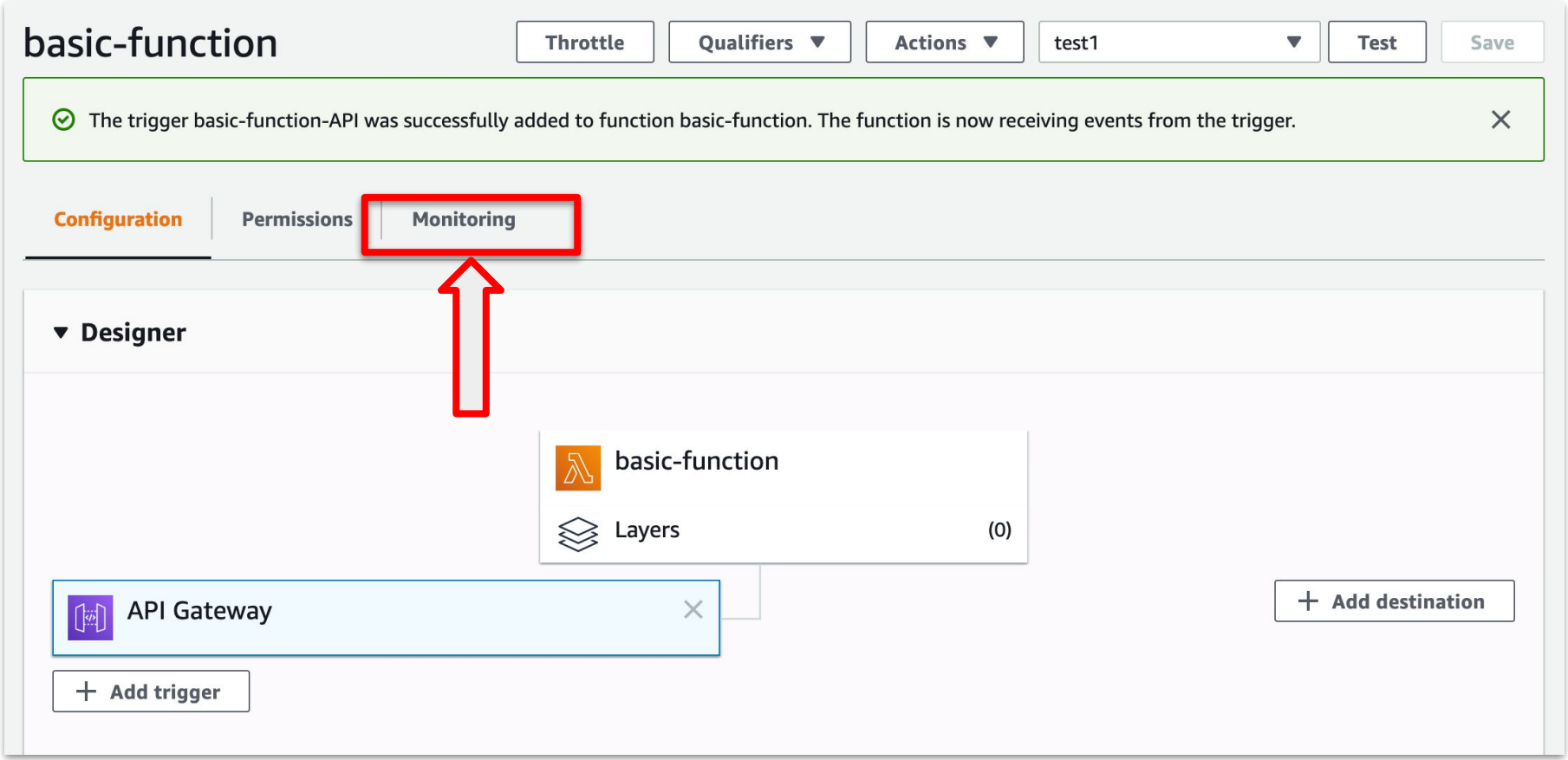
Running lambda via API Gateway

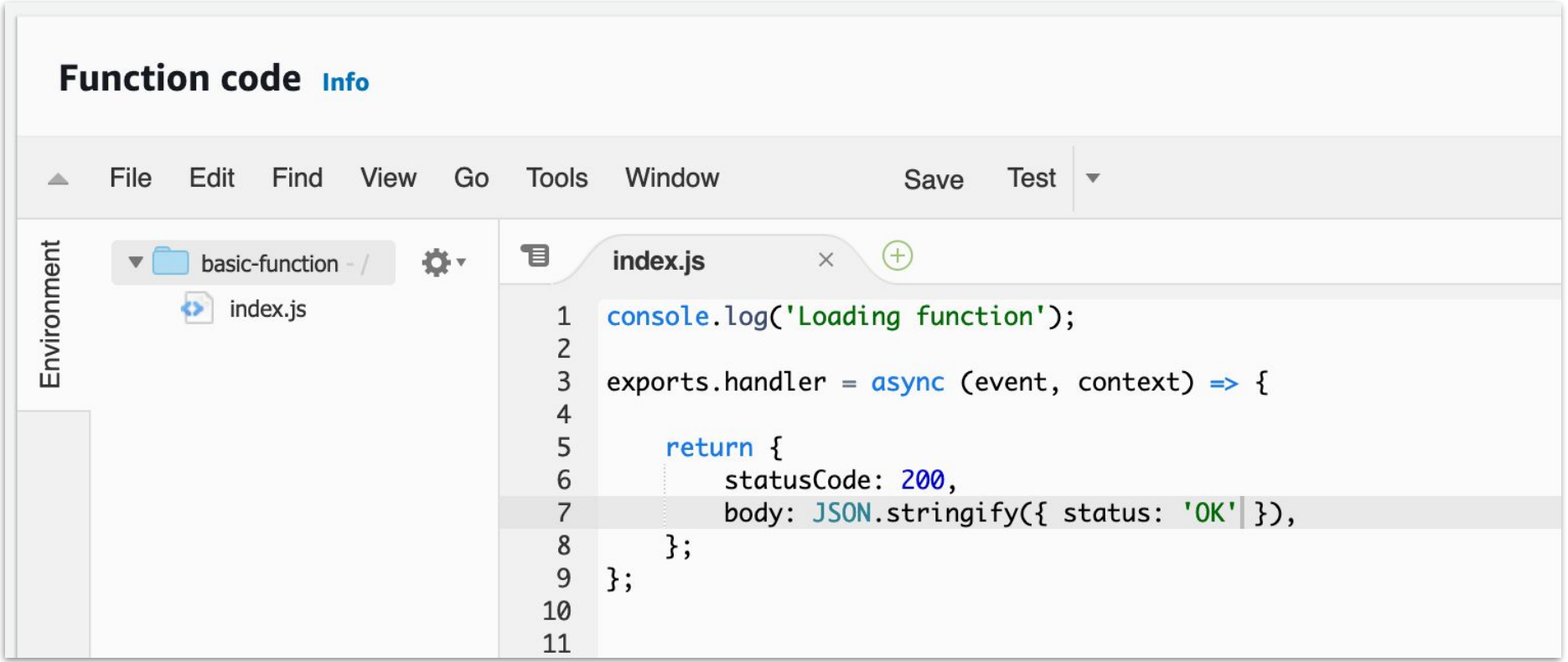

Lambda
Functions
basic-function
ARN - arn:aws:lambda:us-east-1:930272359961:function:basic-function
basic-function
Throttle
Qualifiers
Actions
test1
Test
Save
Configuration
Permissions
Monitoring
Designer
basic-function
Layers
(0)
Add trigger
Add destination

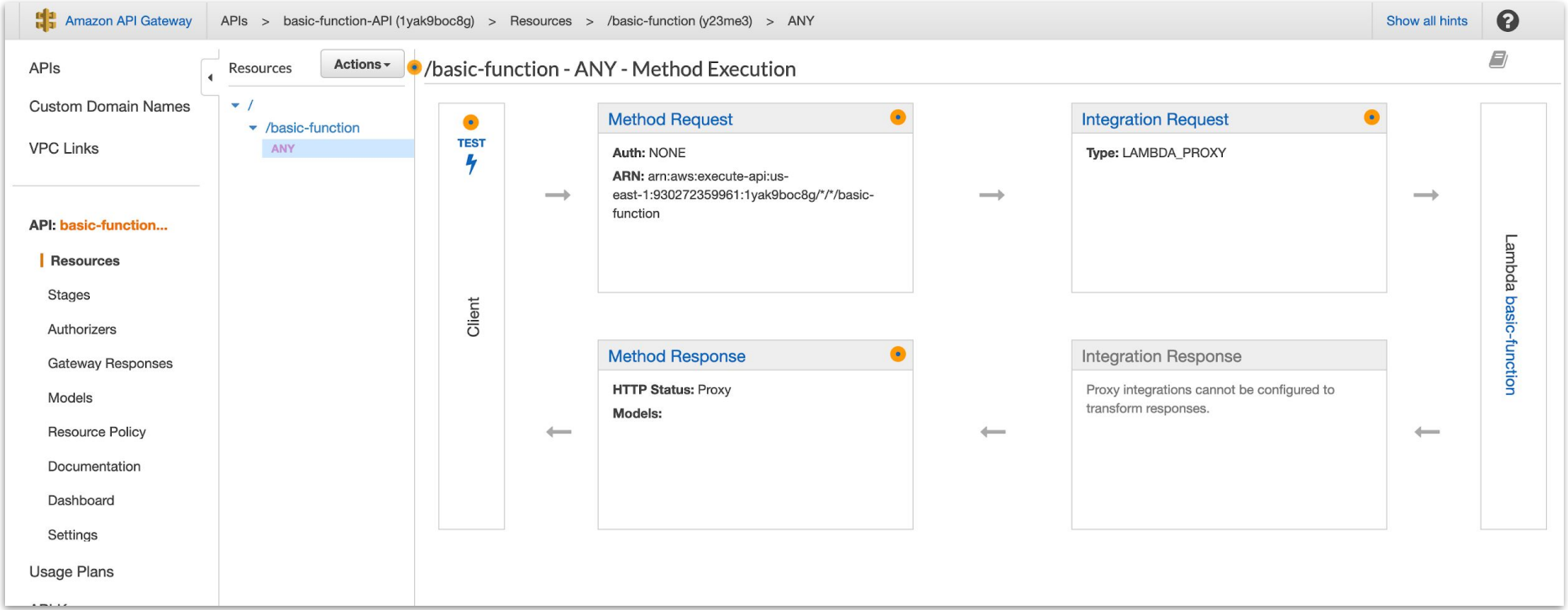

Lambda
Add trigger
Add trigger
Trigger configuration
API Gateway
api application-services aws serverless
Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. Learn more
API
Create a new API or attach an existing one
Create an API
API type
HTTP API
Create an HTTP API.
REST API
Create a REST API.
Security
Configure the security mechanism for your API endpoint.
Open
Don't add any authorization or authentication requirements. Any user can invoke your function with an HTTP call.
Additional settings
Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger.
Learn more about the Lambda permissions model.
Cancel
Add


basic-function
Throttle
Qualifiers
Actions
test1
Test
Save
The trigger basic-function-API was successfully added to function basic-function. The function is now receiving events from the trigger.
Configuration
Permissions
Monitoring
Designer
basic-function
Layers
(0)
API Gateway
Add destination
Add trigger
API Gateway
basic-function-API
arn:aws:execute-api:us-east-1:930272359961:1yak9boc8g/*/*/basic-function
Enabled
Delete
API: api-gateway/1yak9boc8g/*/*/basic-function
API endpoint: https://1yak9boc8g.execute-api.us-east-1.amazonaws.com/default/basic-function
API name: basic-function-API


https://1yak9boc8g.execute-api.us-east-1.amazonaws.com/default/basic-function
200%
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
message: "Internal server error"


basic-function
Throttle
Qualifiers
Actions
test1
Test
Save
The trigger basic-function-API was successfully added to function basic-function. The function is now receiving events from the trigger.
Configuration
Permissions
Monitoring
Designer
basic-function
Layers
(0)
API Gateway
Add destination
Add trigger


Function code Info
File Edit Find View Go Tools Window Save Test
Environment
basic-function - /
index.js
index.js
1 console.log('Loading function');
2
3 exports.handler = async (event, context) => {
4
5 return {
6 statusCode: 200,
7 body: JSON.stringify({ status: 'OK' }),
8 };
9 };
10
11


Amazon API Gateway
APIs > basic-function-API (1yak9boc8g) > Resources > /basic-function (y23me3) > ANY
Show all hints
APIs
Custom Domain Names
VPC Links
API: basic-function...
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Dashboard
Settings
Usage Plans
Resources
Actions
/
/basic-function
ANY
/basic-function - ANY - Method Execution
TEST
Client
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:930272359961:1yak9boc8g/*/*/basic-function
Integration Request
Type: LAMBDA_PROXY
Lambda basic-function
Method Response
HTTP Status: Proxy
Models:
Integration Response
Proxy integrations cannot be configured to transform responses.

Serverless framework

Serverless framework

https://www.serverless.com/

Serverless framework

Why use it?

“The Serverless Framework gives you everything you need to develop, deploy, monitor and secure serverless applications on any cloud.”

Serverless framework

Cloud providers: AWS, Azure, GCP & more
(kubernetes via kubeless!)

Languages: (for AWS Lambda)

- **Node**
- Ruby
- Python
- Go
- Java
- C#
- Custom lambda layers runtimes!

Features:

- Easy deployment
- Local testing
- Plug-ins
- For AWS supports custom cloudformation

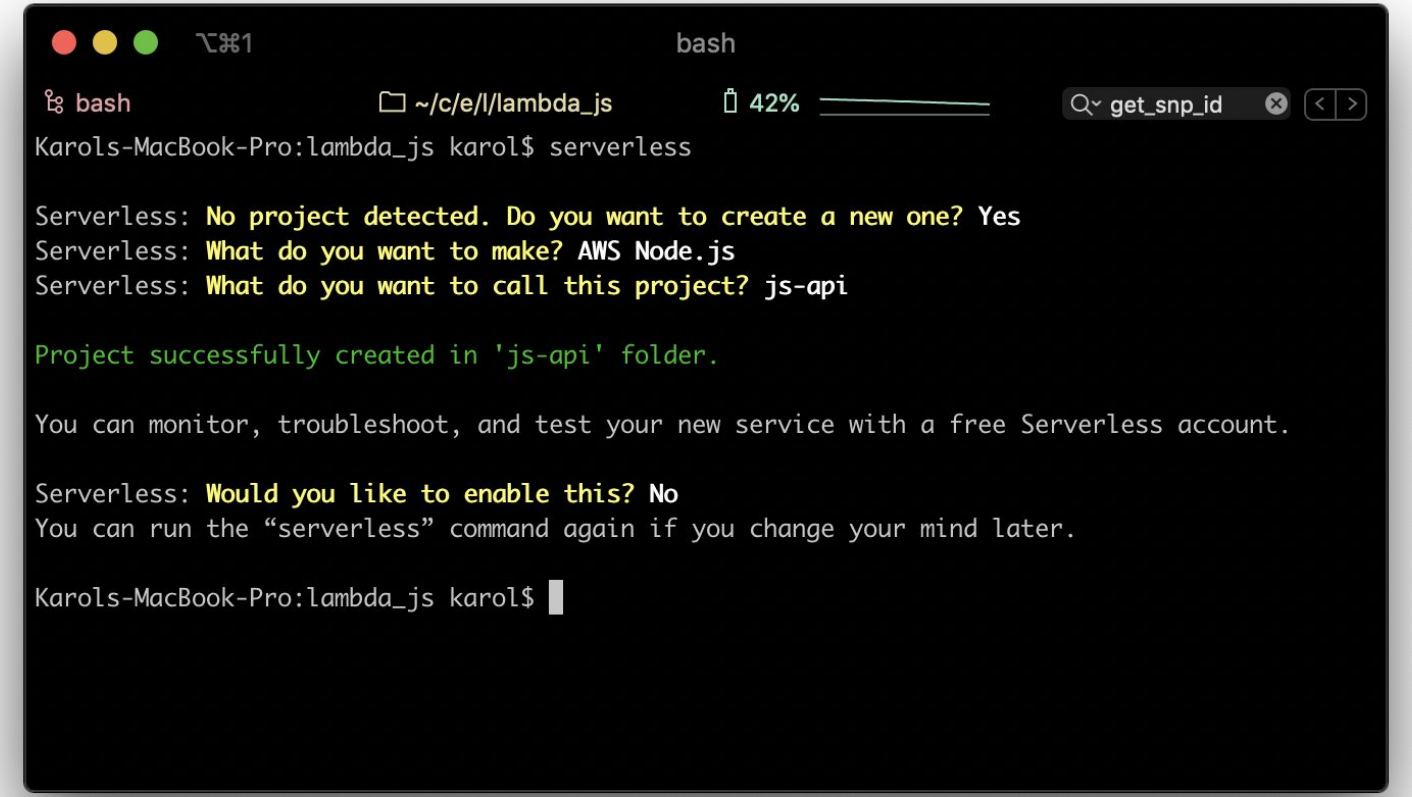
Lambda pricing

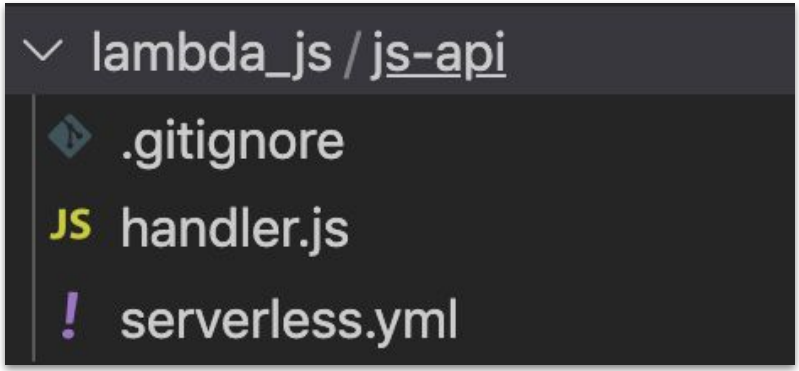
You get quite a lot in the free tier.

Aspect	Pricing	Comment
Requests	$0.20 per 1M requests	The free tier includes 1M requests.
Function memory and run time	$0.0000166667 per GB-second	The free tier includes 400,000 GB-seconds.

Setup

```
npm install serverless -g

serverless
```


⌥⌘1
bash
bash
~/c/e/l/lambda_js
42%
get_snp_id
Karols-MacBook-Pro:lambda_js karol$ serverless
Serverless: No project detected. Do you want to create a new one? Yes
Serverless: What do you want to make? AWS Node.js
Serverless: What do you want to call this project? js-api
Project successfully created in 'js-api' folder.
You can monitor, troubleshoot, and test your new service with a free Serverless account.
Serverless: Would you like to enable this? No
You can run the “serverless” command again if you change your mind later.
Karols-MacBook-Pro:lambda_js karol$



serverless.yml

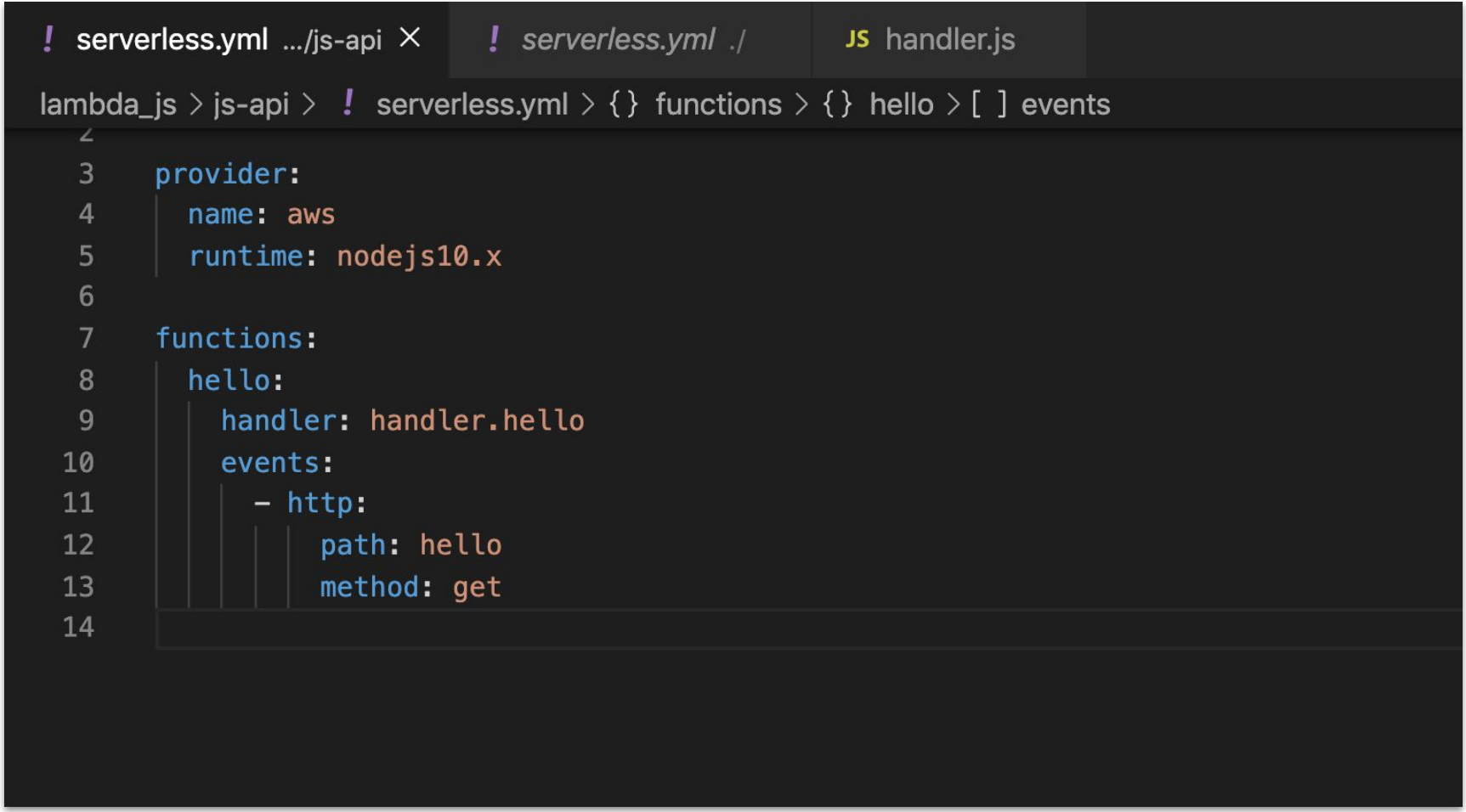
serverless.yml > {} provider

```yaml
service: js-api

provider:
  name: aws
  runtime: nodejs10.x

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

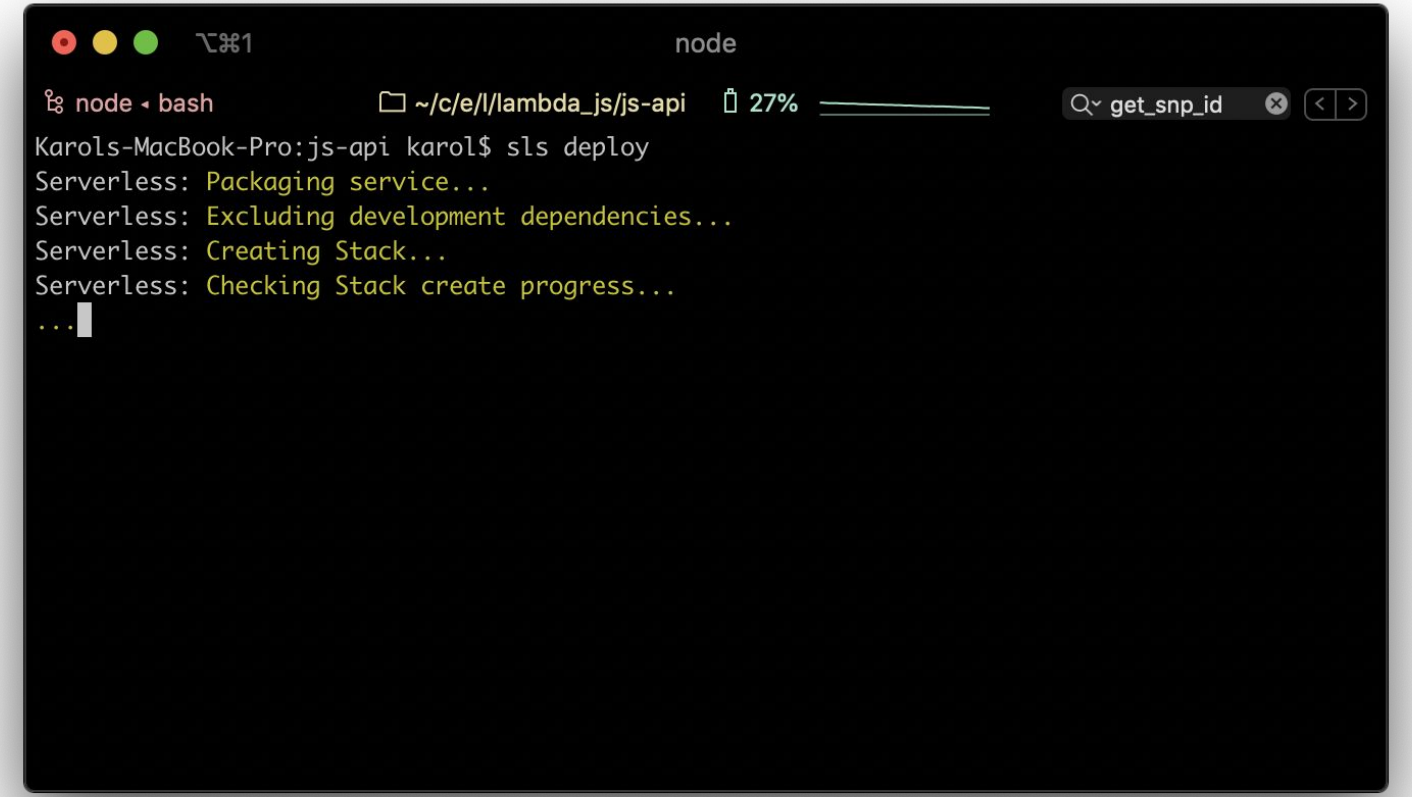
JS handler.js ×

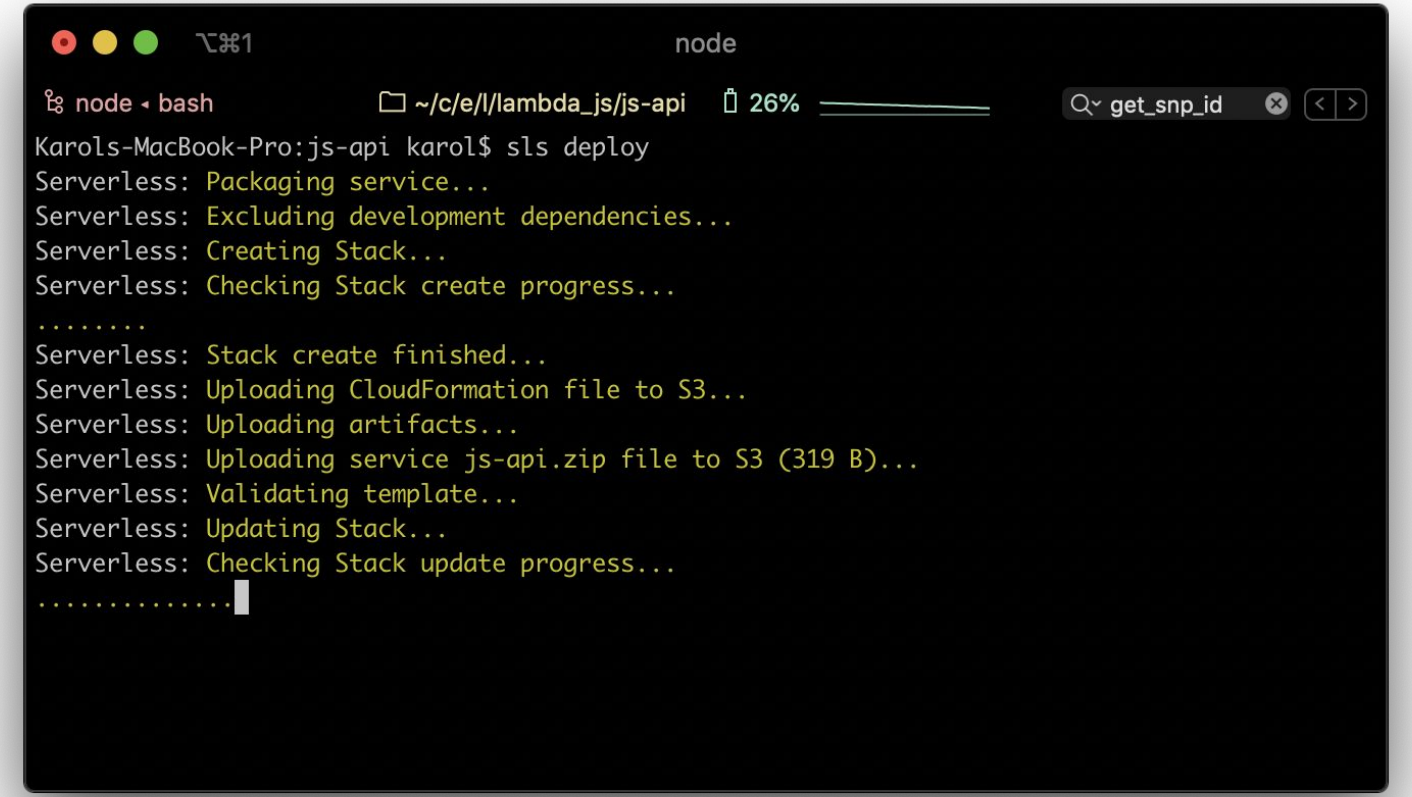
lambda_js > js-api > JS handler.js > ...

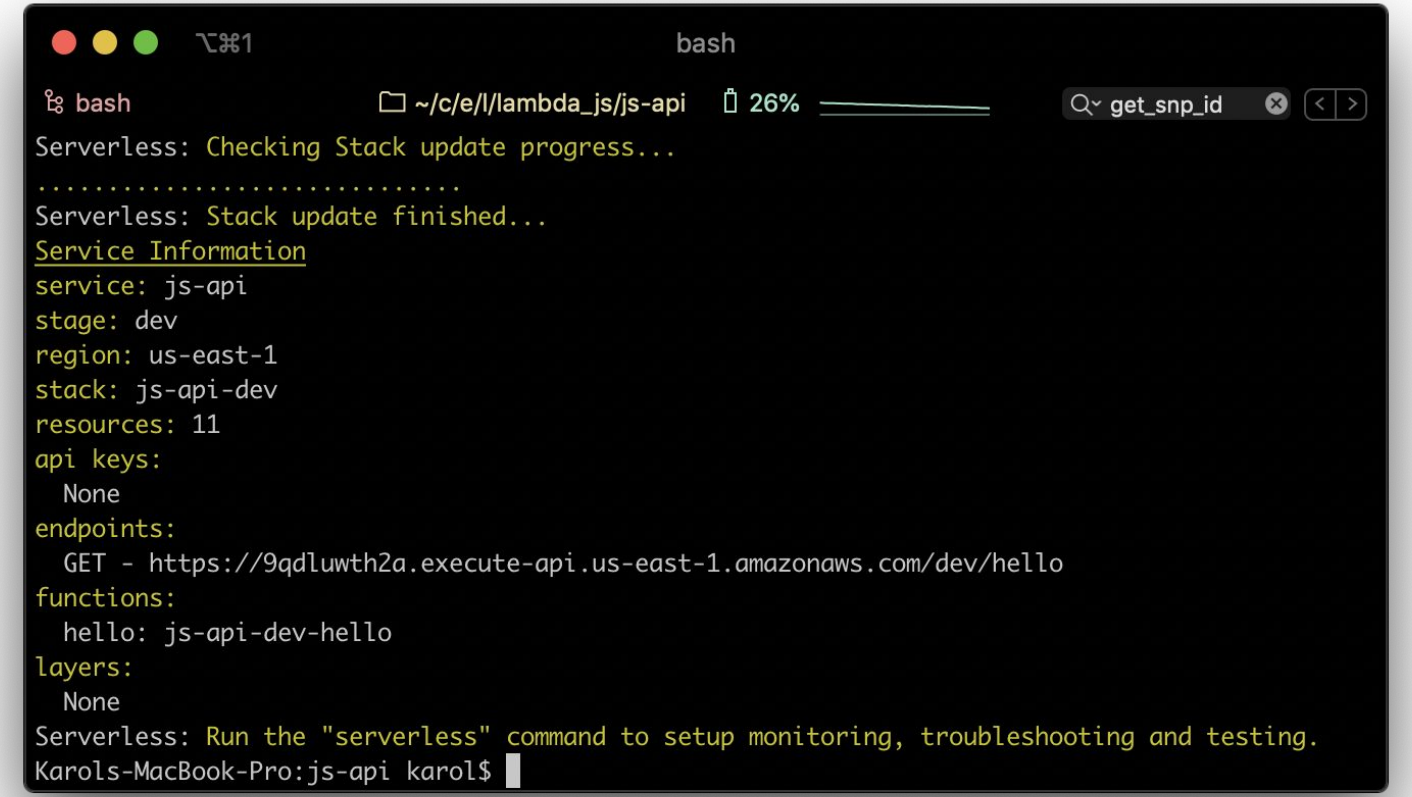
```
1   'use strict';
2
3   module.exports.hello = async event => {
4     return {
5       statusCode: 200,
6       body: JSON.stringify(
7         {
8           message: 'Go Serverless v1.0! Your function executed successfully!',
9           input: event,
10        },
11        null,
12        2
13      ),
14    };
15  };
16
```


! serverless.yml .../js-api
! serverless.yml ./
JS handler.js
lambda_js > js-api > ! serverless.yml > { } functions > { } hello > [ ] events
provider:
name: aws
runtime: nodejs10.x
functions:
hello:
handler: handler.hello
events:
- http:
path: hello
method: get


⌥⌘1
bash
bash
~/c/e/l/lambda_js
27%
get_snp_id
Karols-MacBook-Pro:lambda_js karol$ sls deploy

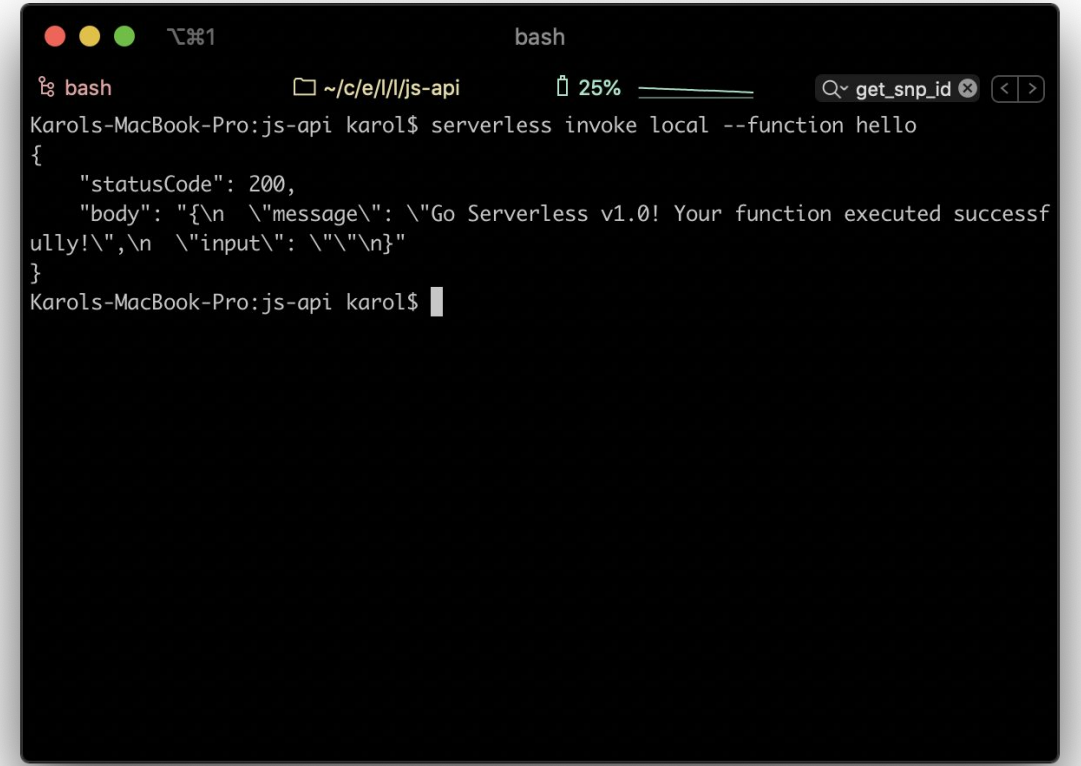

⌥⌘1
node
node ◂ bash
~/c/e/l/lambda_js/js-api
27%
get_snp_id
Karols-MacBook-Pro:js-api karol$ sls deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
...


⌥⌘1
node
node ◂ bash
~/c/e/l/lambda_js/js-api
26%
get_snp_id
Karols-MacBook-Pro:js-api karol$ sls deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
.....
Serverless: Stack create finished...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service js-api.zip file to S3 (319 B)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.....

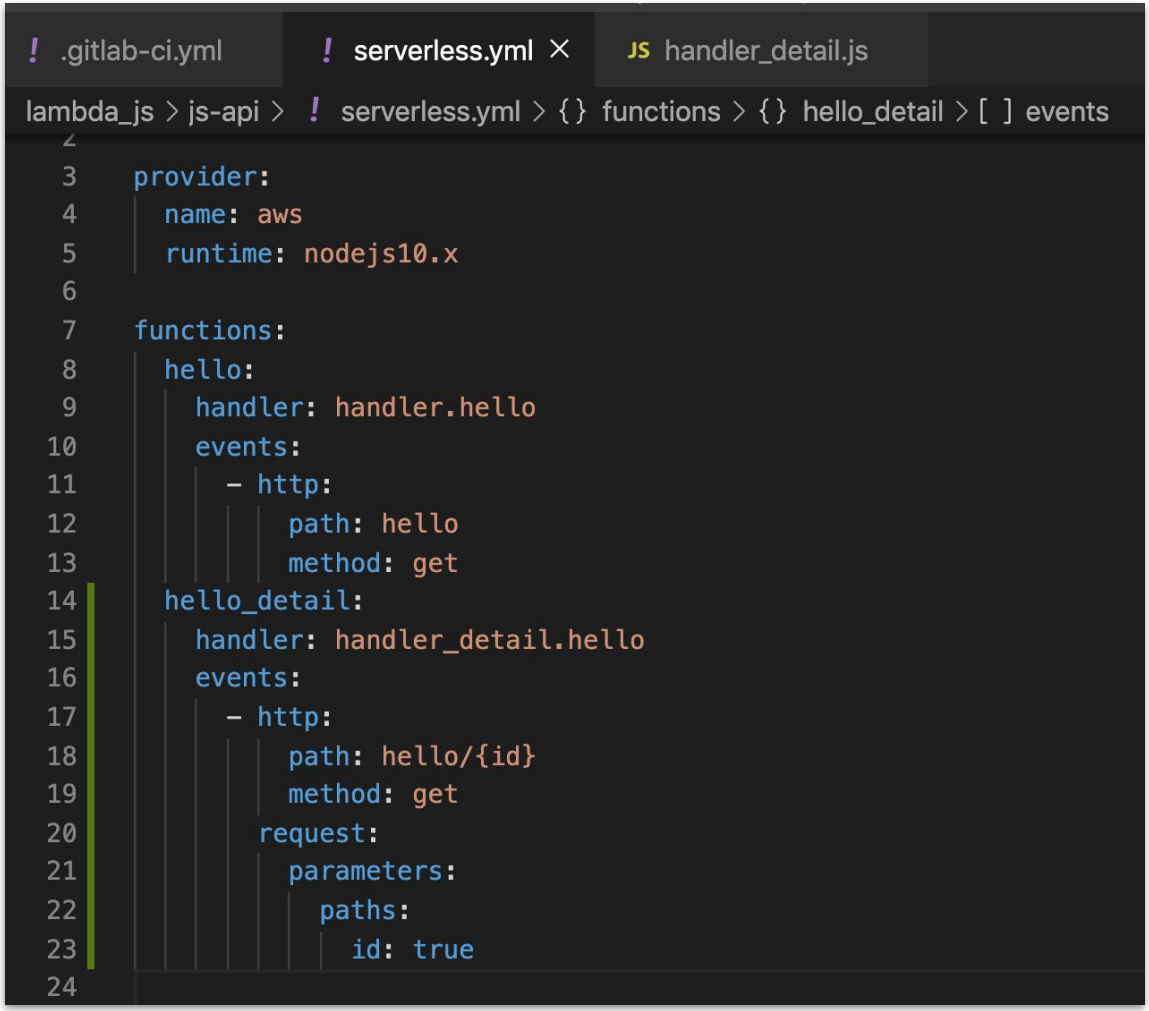

⌥⌘1
bash
bash
~/c/e/l/lambda_js/js-api
26%
get_snp_id
Serverless: Checking Stack update progress...
.....
Serverless: Stack update finished...
Service Information
service: js-api
stage: dev
region: us-east-1
stack: js-api-dev
resources: 11
api keys:
None
endpoints:
GET - https://9qdluwth2a.execute-api.us-east-1.amazonaws.com/dev/hello
functions:
hello: js-api-dev-hello
layers:
None
Serverless: Run the "serverless" command to setup monitoring, troubleshooting and testing.
Karols-MacBook-Pro:js-api karol$

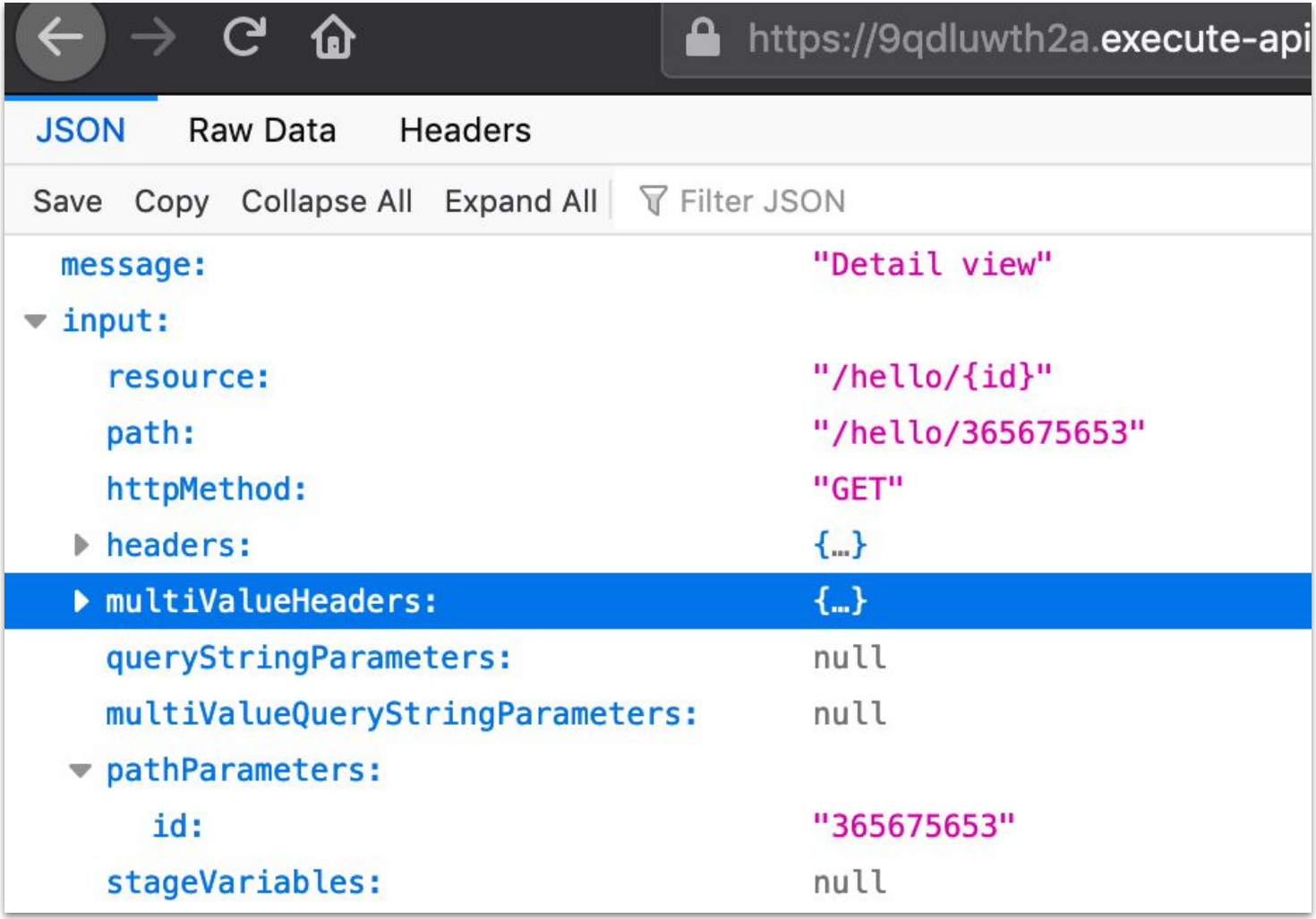

https://9qdluwth2a.execute-api.us-east-1.amazonaws.com/dev/hello
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
message: "Go Serverless v1.0! Your function executed successfully!"
input: {…}

Local testing



https://www.serverless.com/framework/docs/providers/aws/cli-reference/invoke-local/#aws---invoke-local


! .gitlab-ci.yml
! serverless.yml
JS handler_detail.js
lambda_js > js-api > ! serverless.yml > {} functions > {} hello_detail > [ ] events
provider:
name: aws
runtime: nodejs10.x
functions:
hello:
handler: handler.hello
events:
- http:
path: hello
method: get
hello_detail:
handler: handler_detail.hello
events:
- http:
path: hello/{id}
method: get
request:
parameters:
paths:
id: true


https://9qdluwth2a.execute-api
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
message: "Detail view"
input:
resource: "/hello/{id}"
path: "/hello/365675653"
httpMethod: "GET"
headers: {…}
multiValueHeaders: {…}
queryStringParameters: null
multiValueQueryStringParameters: null
pathParameters:
id: "365675653"
stageVariables: null

Development workflow

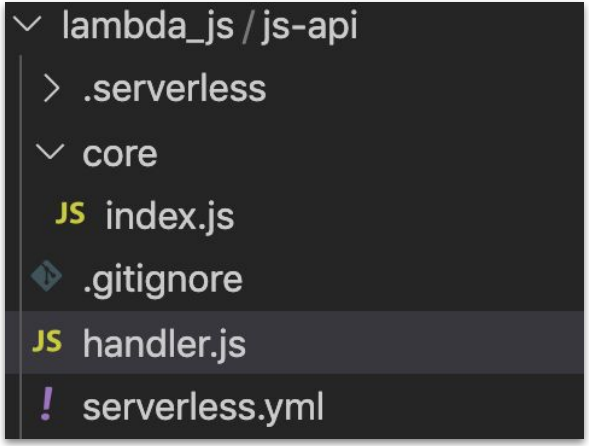
1. Write your functions
2. Use `serverless deploy` only when you've made changes to serverless.yml and in CI/CD systems. For more information on setting up CI/CD for your Serverless app, read this article.
3. Use `serverless deploy function -f myFunction` to rapidly deploy changes when you are working on a specific AWS Lambda Function.
4. Use `serverless invoke -f myFunction -l` to test your AWS Lambda Functions on AWS.
5. Open up a separate tab in your console and stream logs in there via `serverless logs -f myFunction -t.`
6. Write tests to run locally.

Testing

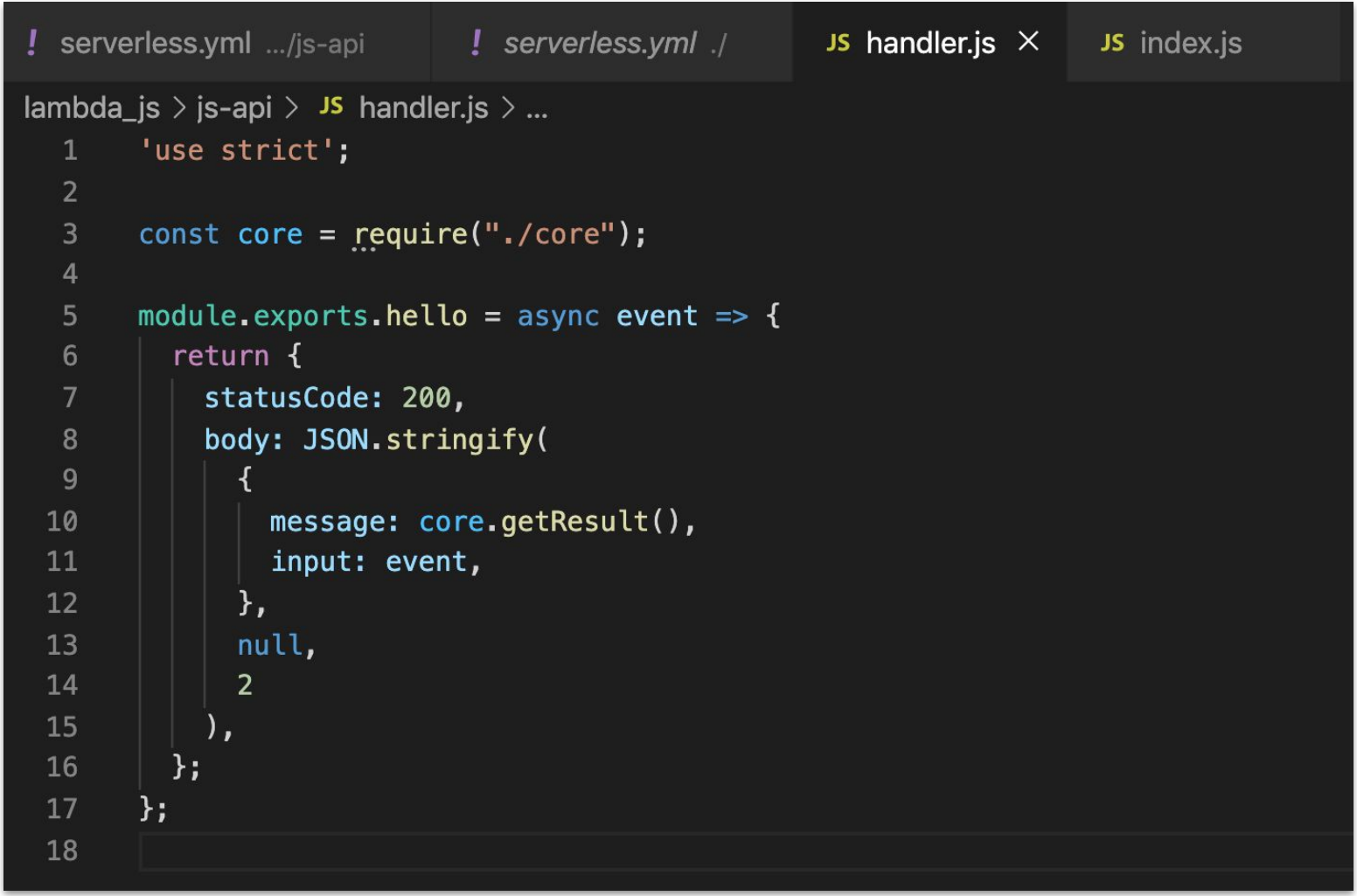
Testing

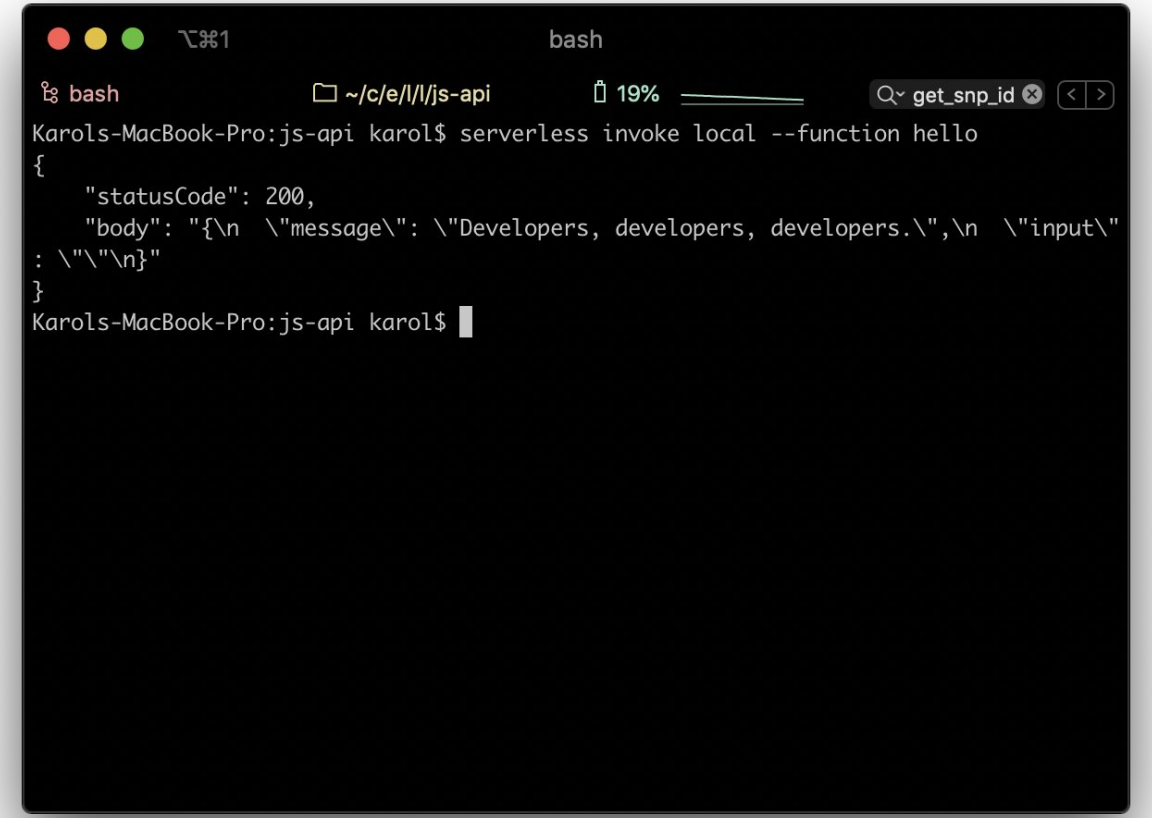
- Write your business logic so that it is separate from your FaaS provider (e.g., AWS Lambda), to keep it provider-independent, reusable and more easily testable.
- When your business logic is written separately from the FaaS provider, you can write traditional Unit Tests to ensure it is working properly.
- Write Integration Tests to verify integrations with other services are working correctly.

Core library


lambda_js / js-api
.serverless
core
index.js
.gitignore
handler.js
serverless.yml


serverless.yml .../js-api
serverless.yml ./
handler.js
index.js
lambda_js > js-api > core > index.js > ...
1 'use strict';
2
3 module.exports.getResult = () => {
4 return "Developers, developers, developers."
5 };
6

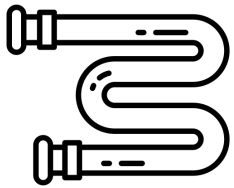

! serverless.yml .../js-api
! serverless.yml ./
JS handler.js
JS index.js
lambda_js > js-api > JS handler.js > ...
1 'use strict';
2
3 const core = require("./core");
4
5 module.exports.hello = async event => {
6 return {
7 statusCode: 200,
8 body: JSON.stringify(
9 {
10 message: core.getResult(),
11 input: event,
12 },
13 null,
14 2
15),
16 };
17 };
18


⌥⌘1
bash
bash
~/c/e/l/l/js-api
19%
get_snp_id
Karols-MacBook-Pro:js-api karol$ serverless invoke local --function hello
{
"statusCode": 200,
"body": "{\n \"message\": \"Developers, developers, developers.\",\n \"input\"
: \"\"\n}"
}
Karols-MacBook-Pro:js-api karol$

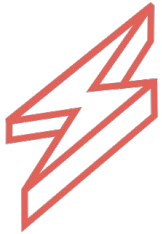
CI/CD

CI/CD

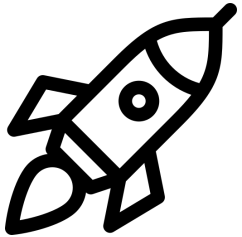
1. Push to repo



2. Trigger CI/CD pipeline



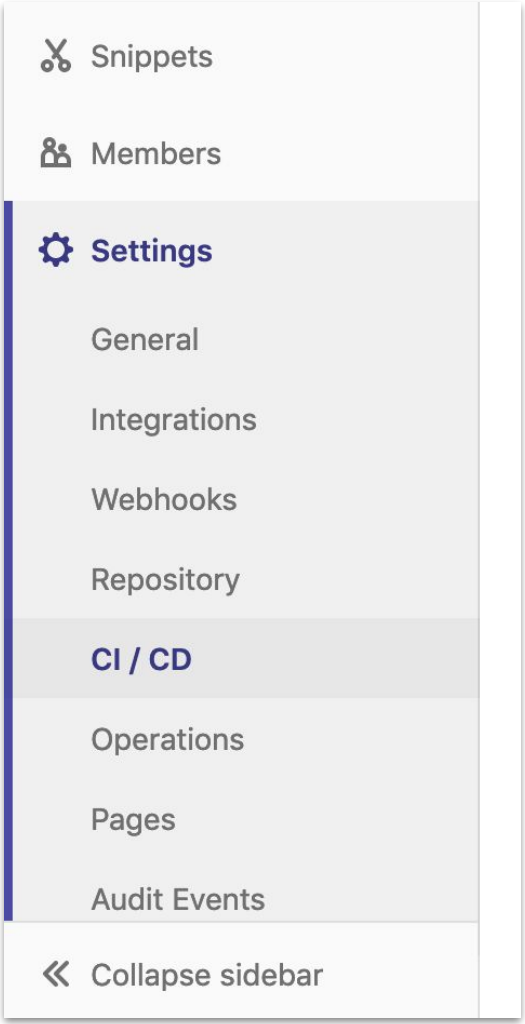
3. Install sls in node container



4. Deploy!


! .gitlab-ci.yml ×
lambda_js > js-api > ! .gitlab-ci.yml > [ ] stages
1 stages:
2 - deploy_api
3
4 image: node:12
5
6 deploy:
7 stage: deploy_api
8 before_script:
9 - npm install -g serverless
10 script:
11 - serverless deploy --stage ${CI_COMMIT_REF_NAME:-master} -v
12 only:
13 - master
14 - develop
15

Repository settings



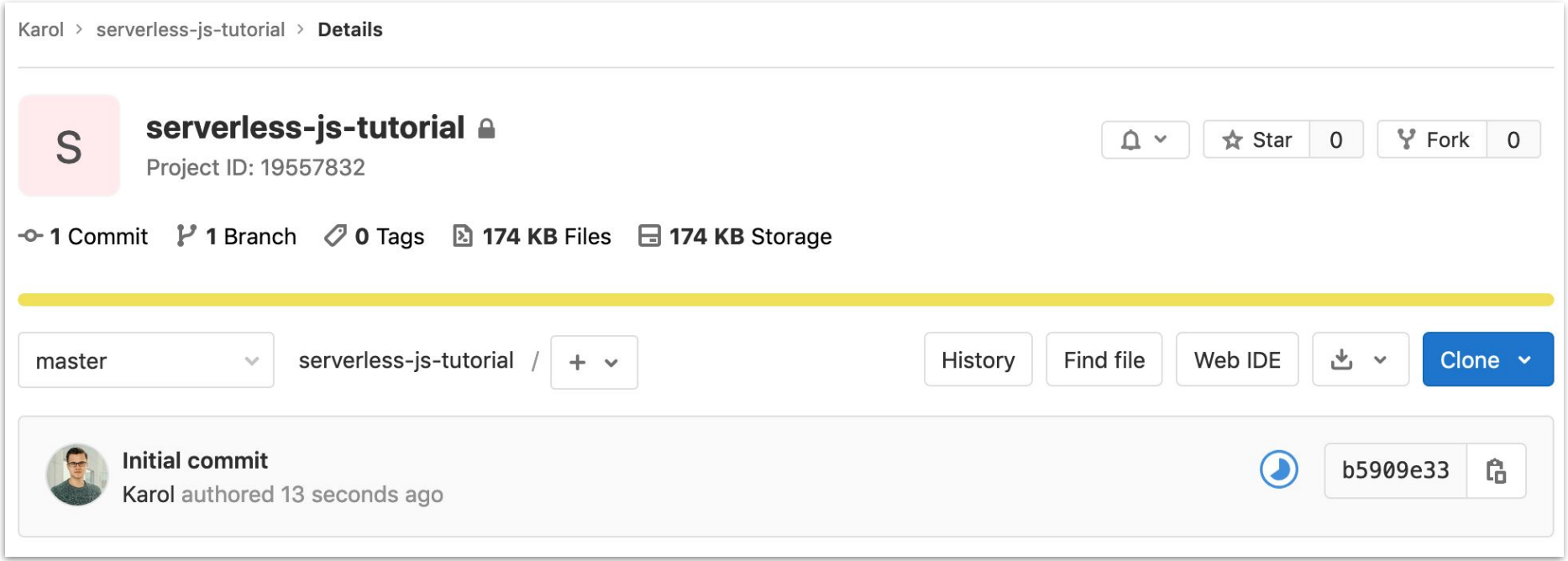
Variables

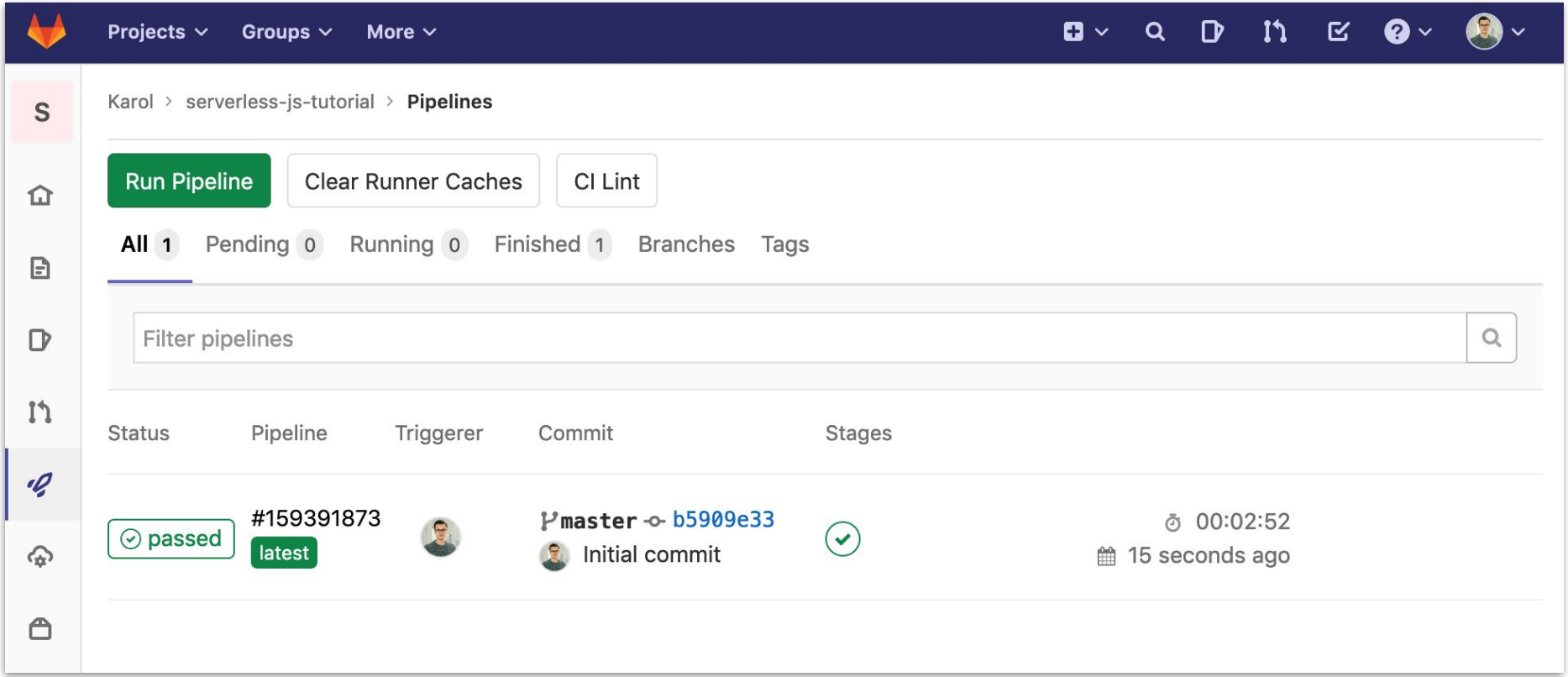
Environment variables are applied to environments via the runner. They can be protected by only exposing them to protected branches or tags. Additionally, they can be masked so they are hidden in job logs, though they must match certain regexp requirements to do so. You can use environment variables for passwords, secret keys, or whatever you want. You may also add variables that are made available to the running application by prepending the variable key with `K8S_SECRET_`. More information

Environment variables are configured by your administrator to be protected by default

Type	↑ Key	Value	Protected	Masked	Environments
Variable	AWS_ACCESS_KEY_ID	**********	✓	✓	All (default)
Variable	AWS_SECRET_ACCESS_KEY	**********	✓	✓	All (default)

Reveal values | Add Variable


Karol > serverless-js-tutorial > Details
S
serverless-js-tutorial
Project ID: 19557832
Star 0
Fork 0
1 Commit
1 Branch
0 Tags
174 KB Files
174 KB Storage
master
serverless-js-tutorial /
History
Find file
Web IDE
Clone
Initial commit
Karol authored 13 seconds ago
b5909e33


Projects
Groups
More
S
Karol > serverless-js-tutorial > Pipelines
Run Pipeline
Clear Runner Caches
CI Lint
All 1
Pending 0
Running 0
Finished 1
Branches
Tags
Filter pipelines
Status
Pipeline
Triggerer
Commit
Stages
passed
#159391873
latest
master b5909e33
Initial commit
00:02:52
15 seconds ago

That’s it!

Resources

Code/presentation
https://gitlab.com/horosin/serverless-js-tutorial

My newsletter
https://mailchi.mp/a1db5198f63f/karol-horosin-conference-newsletter

Serverless Stack Tutorial (including React frontend)
https://serverless-stack.com/

Acknowledgements

Some icons made by Icongeek26