МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

**Звіт**

до лабораторної роботи №1

**Зберігання даних у DynamoDB та створення Lambda функцій**

Виконав:

Студент гр. ІТ-11сп

Олег КУЗЬО

|  |  |
| --- | --- |
| **Балів** | **Дата** |
|  |  |

Прийняв:

Доц. каф. Михайло ЛУЧКЕВИЧ

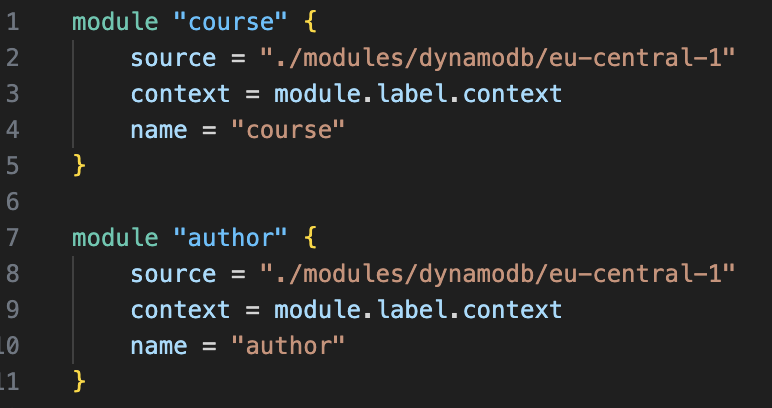
Львів-2023

Лабораторна робота №1

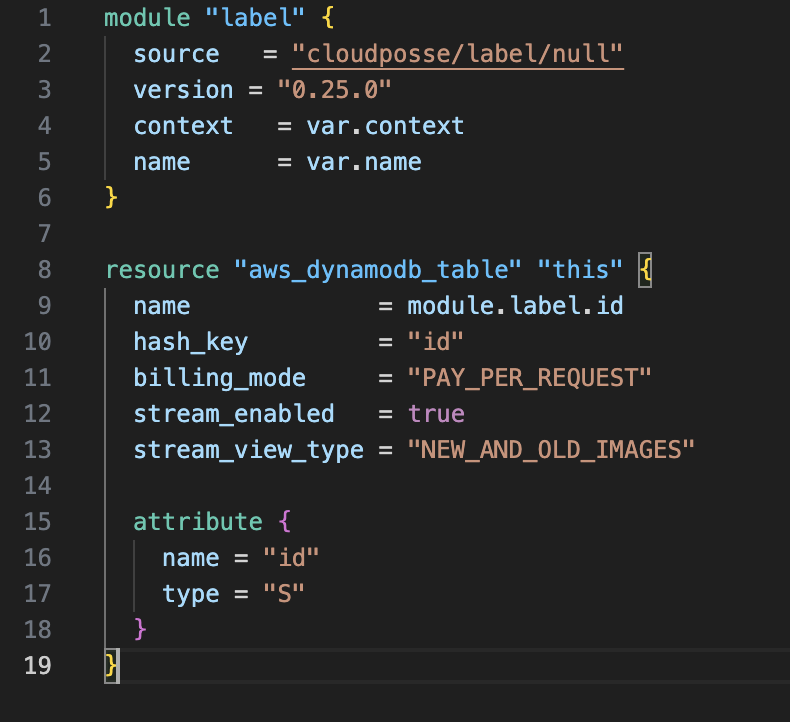
**Мета роботи:** Зберігання даних у DynamoDB та створення lambda функцій

**Хід роботи**

За допомогою terraform в main.tf описуємо дві таблиці: authors I courses:



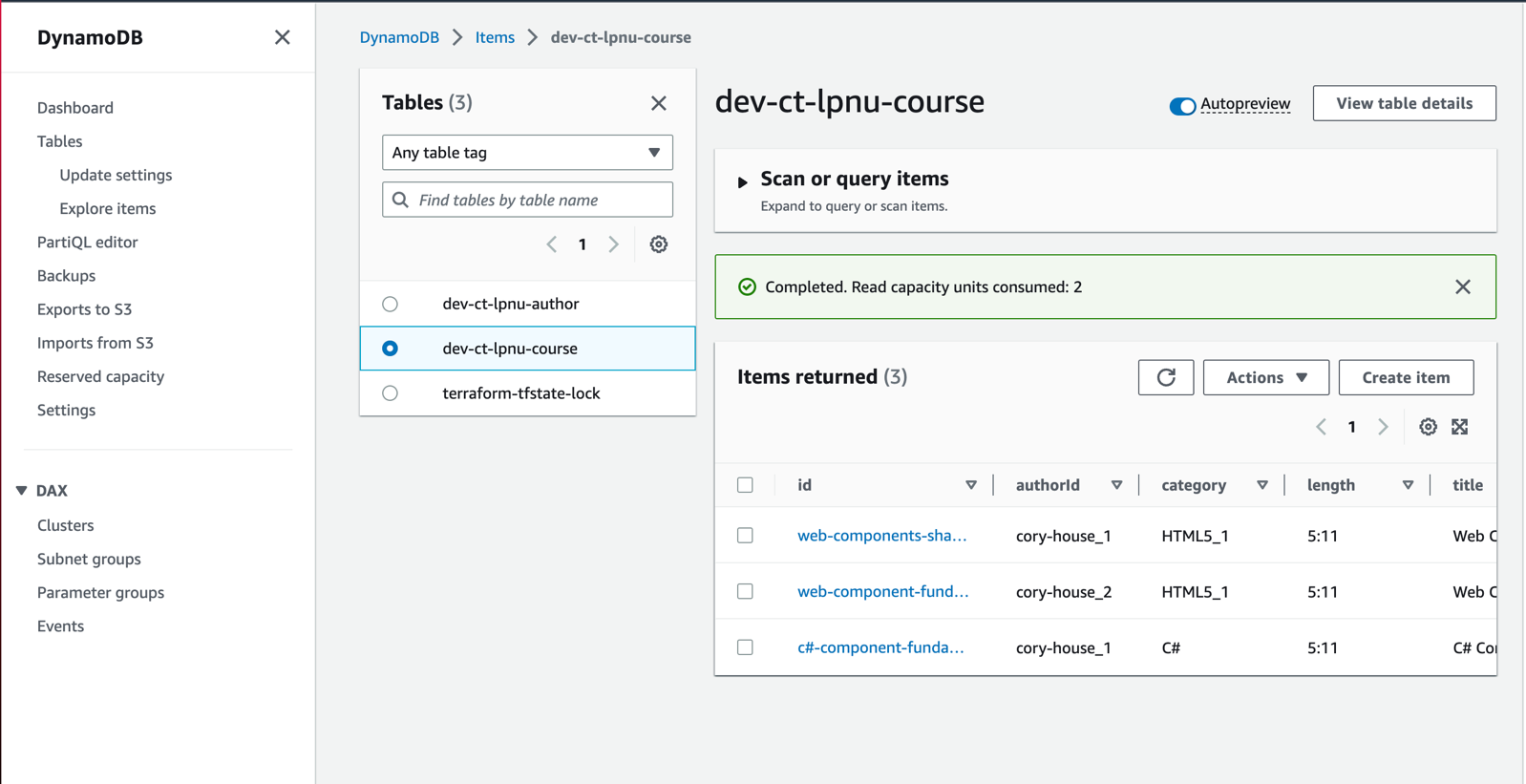
В DynamoDB/eu-central-1/main.tf прописуємо наступний код:

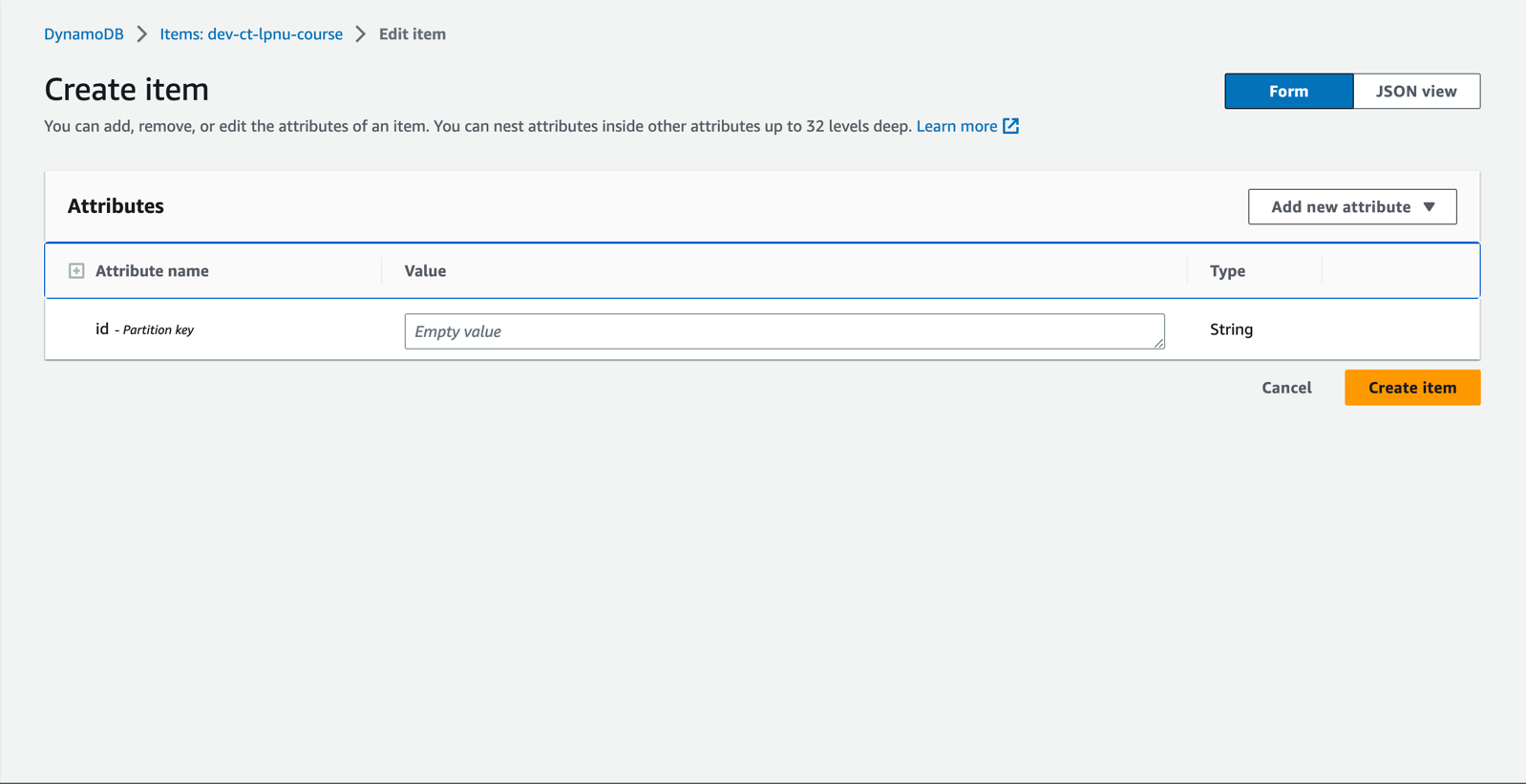


В результаті на вкладці DynamoDB будуть знаходитися дві таблиці:  

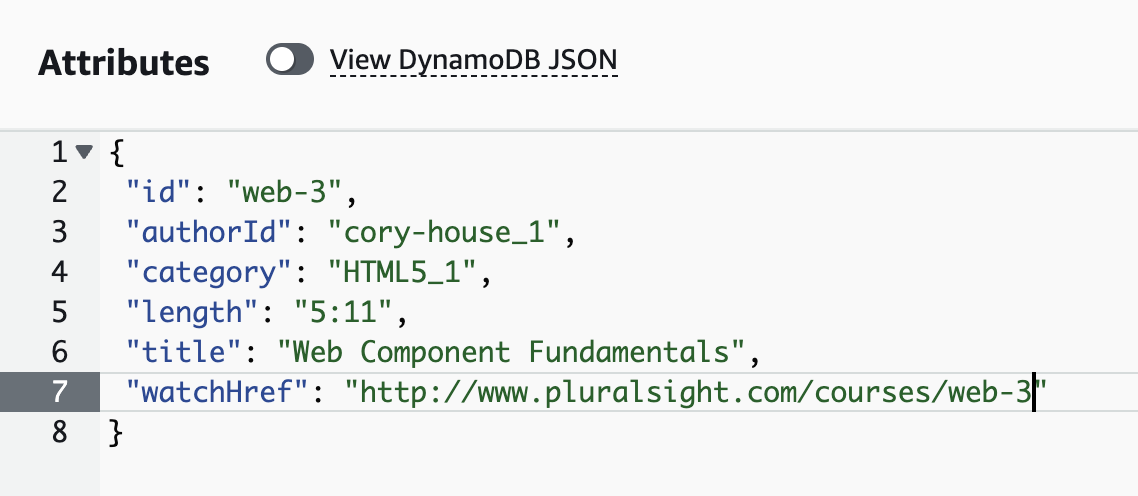

Для того, щоб добавити запис в таблицю використати наступний метод:

На вкладці Explore Items обираємо необхідну таблицю, після чого на екрані появиться вкладка з інформацією про наявні записи в таблиці:

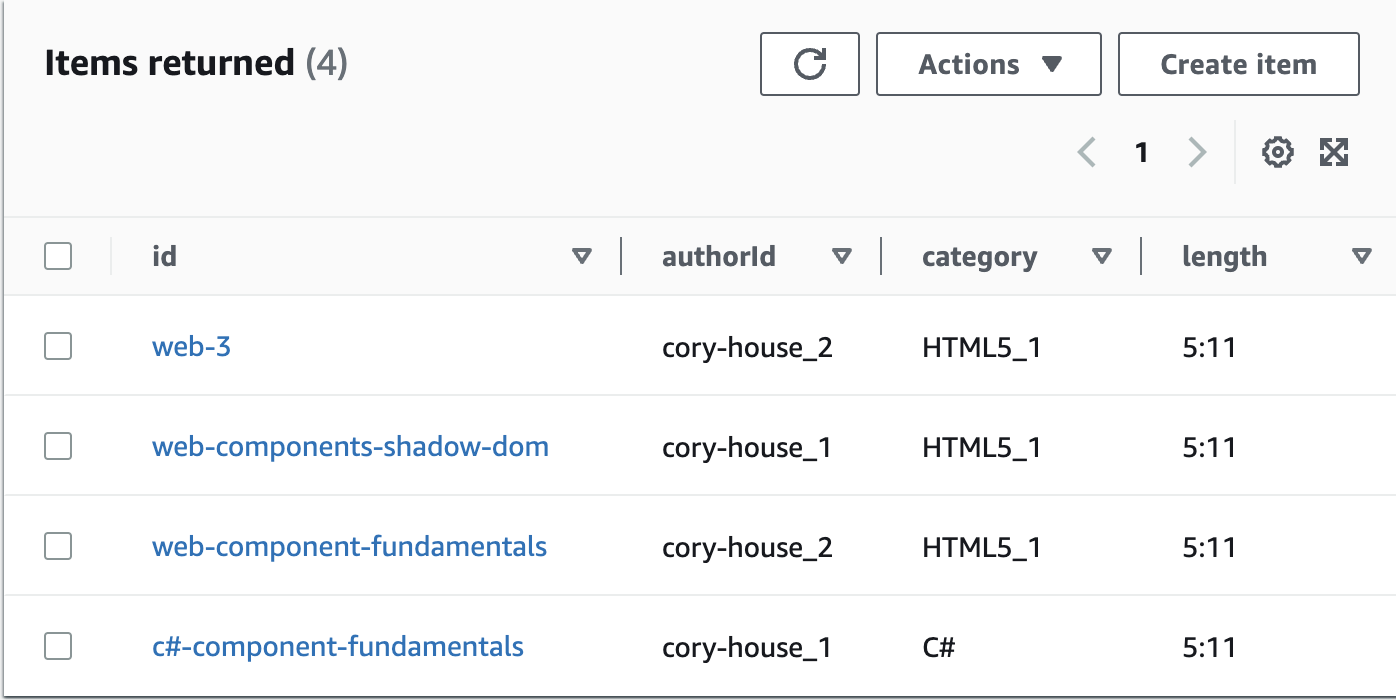
Натискаємо на кнопку Create item, на екрані появиться вікно для створення запису. Обираємо JSON view:



На екрані появиться поле, де необхідно вписати запис як JSON-файл. Вимикаємо View DynamoDB JSON і вводимо запис, після чого натискаємо Create Item:



В результаті на вкладці Explore items у вибраній таблиці буде новий запис:



Далі створимо лямбда-функції, які будуть використовувати ці таблиці.

Для цієї лабораторної роботи було створено 6 lambda-функцій:

* get-all-authors щоб повернути всіх користувачів у базі даних
* get-all-courses щоб повернути всі курси в базі даних
* get-course повернути лише один курс
* save-course створити новий курс
* update-course оновити курс
* delete-course щоб видалити курс

Для того, щоб лямбда функції правильно функціонували, необхідно створити наступні файли: main.tf, variables.tf, outputs.tf

main.tf

module "label" {

source = "cloudposse/label/null"

version = "0.25.0"

context = var.context

name = var.name

}

module "label\_courses" {

source = "cloudposse/label/null"

version = "0.25.0"

context = var.context

name = var.name\_courses

}

module "lambda" {

source = "terraform-aws-modules/lambda/aws"

version = "4.13.0"

function\_name = module.label.id

description = "Get all authors"

handler = "index.handler"

runtime = "nodejs12.x"

source\_path = "${path.module}/lambda\_src/get\_all\_authors/index.js"

environment\_variables = {

TABLE\_NAME = var.table\_author\_name

}

attach\_policy = true

policy = "arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess"

attach\_policies = true

policies = ["arn:aws:iam::aws:policy/AWSXrayReadOnlyAccess"]

number\_of\_policies = 1

attach\_policy\_statements = true

policy\_statements = {

dynamodb = {

effect = "Allow",

actions = ["dynamodb:Scan"],

resources = [var.table\_author\_arn]

}

# s3\_read = {

# effect = "Deny",

# actions = ["s3:HeadObject", "s3:GetObject"],

# resources = ["arn:aws:s3:::my-bucket/\*"]

# }

}

tags = module.label.tags

}

module "lambda\_courses" {

source = "terraform-aws-modules/lambda/aws"

version = "4.13.0"

function\_name = module.label\_courses.id

description = "Get all courses"

handler = "index.handler"

runtime = "nodejs12.x"

source\_path = "${path.module}/lambda\_src/get\_all\_courses/index.js"

environment\_variables = {

TABLE\_NAME = var.table\_courses\_name

}

attach\_policy = true

policy = "arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess"

attach\_policies = true

policies = ["arn:aws:iam::aws:policy/AWSXrayReadOnlyAccess"]

number\_of\_policies = 1

attach\_policy\_statements = true

policy\_statements = {

dynamodb = {

effect = "Allow",

actions = ["dynamodb:Scan"],

resources = [var.table\_courses\_arn]

}

# s3\_read = {

# effect = "Deny",

# actions = ["s3:HeadObject", "s3:GetObject"],

# resources = ["arn:aws:s3:::my-bucket/\*"]

# }

}

tags = module.label\_courses.tags

}

module "label\_get\_course" {

source = "cloudposse/label/null"

version = "0.25.0"

context = var.context

name = var.name\_get\_course

}

module "label\_save\_course" {

source = "cloudposse/label/null"

version = "0.25.0"

context = var.context

name = var.name\_save\_course

}

module "label\_update\_course" {

source = "cloudposse/label/null"

version = "0.25.0"

context = var.context

name = var.name\_update\_course

}

module "label\_delete\_course" {

source = "cloudposse/label/null"

version = "0.25.0"

context = var.context

name = var.name\_delete\_course

}

module "lambda\_get\_course" {

source = "terraform-aws-modules/lambda/aws"

version = "4.13.0"

function\_name = module.label\_get\_course.id

description = "Get course"

handler = "index.handler"

runtime = "nodejs12.x"

source\_path = "${path.module}/lambda\_src/get\_course/index.js"

environment\_variables = {

TABLE\_NAME = var.table\_courses\_name

}

create\_role = false

lambda\_role = var.lambda\_courses\_role\_arn

tags = module.label\_get\_course.tags

}

module "lambda\_save\_course" {

source = "terraform-aws-modules/lambda/aws"

version = "4.13.0"

function\_name = module.label\_save\_course.id

description = "Save course"

handler = "index.handler"

runtime = "nodejs12.x"

source\_path = "${path.module}/lambda\_src/save\_course/index.js"

environment\_variables = {

TABLE\_NAME = var.table\_courses\_name

}

create\_role = false

lambda\_role = var.lambda\_courses\_role\_arn

tags = module.label\_save\_course.tags

}

module "lambda\_update\_course" {

source = "terraform-aws-modules/lambda/aws"

version = "4.13.0"

function\_name = module.label\_update\_course.id

description = "Update course"

handler = "index.handler"

runtime = "nodejs12.x"

source\_path = "${path.module}/lambda\_src/update\_course/index.js"

environment\_variables = {

TABLE\_NAME = var.table\_courses\_name

}

create\_role = false

lambda\_role = var.lambda\_courses\_role\_arn

tags = module.label\_update\_course.tags

}

module "lambda\_delete\_course" {

source = "terraform-aws-modules/lambda/aws"

version = "4.13.0"

function\_name = module.label\_delete\_course.id

description = "Delete course"

handler = "index.handler"

runtime = "nodejs12.x"

source\_path = "${path.module}/lambda\_src/delete\_course/index.js"

environment\_variables = {

TABLE\_NAME = var.table\_courses\_name

}

create\_role = false

lambda\_role = var.lambda\_courses\_role\_arn

tags = module.label\_delete\_course.tags

}

Variables.tf

variable "table\_author\_name" {

type = string

}

variable "table\_author\_arn" {

type = string

}

variable "table\_courses\_name" {

type = string

}

variable "table\_courses\_arn" {

type = string

}

variable "name\_courses" {

type = string

}

variable "lambda\_courses\_role\_arn" {

type = string

}

# additional lambda function

variable "name\_get\_course" {

type = string

}

variable "name\_save\_course" {

type = string

}

variable "name\_update\_course" {

type = string

}

variable "name\_delete\_course" {

type = string

}

Outputs.tf

output "lambda\_courses\_invoke\_arn" {

value = module.lambda\_courses.lambda\_function\_invoke\_arn

}

output "lambda\_authors\_invoke\_arn" {

value = module.lambda.lambda\_function\_invoke\_arn

}

output "lambda\_authors\_function\_name" {

value = module.lambda.lambda\_function\_name

}

Також необхідно створити безпосередньо самі lambda функції за допомогою мови програмування javascript

Delete\_cource.js

const AWS = require("aws-sdk");

const dynamodb = new AWS.DynamoDB({

region: process.env.AWS\_REGION,

apiVersion: "2012-08-10"

});

exports.handler = (event, context, callback) => {

const params = {

Key: {

id: {

S: event.id

}

},

TableName: process.env.TABLE\_NAME

};

dynamodb.deleteItem(params, (err, data) => {

if (err) {

console.log(err);

callback(err);

} else {

callback(null, data);

}

});

};

get\_all\_authors

const AWS = require("aws-sdk");

const dynamodb = new AWS.DynamoDB({

region: process.env.AWS\_REGION,

apiVersion: "2012-08-10"

});

exports.handler = (event, context, callback) => {

const params = {

TableName: process.env.TABLE\_NAME

};

dynamodb.scan(params, (err, data) => {

if (err) {

console.log(err);

callback(err);

} else {

callback(null, data);

}

});

};

get\_all\_courses

const AWS = require("aws-sdk");

const dynamodb = new AWS.DynamoDB({

region: process.env.AWS\_REGION,

apiVersion: "2012-08-10"

});

exports.handler = (event, context, callback) => {

const params = {

TableName: process.env.TABLE\_NAME

};

dynamodb.scan(params, (err, data) => {

if (err) {

console.log(err);

callback(err);

} else {

callback(null, data);

}

});

};

get\_course

const AWS = require("aws-sdk");

const dynamodb = new AWS.DynamoDB({

region: process.env.AWS\_REGION,

apiVersion: "2012-08-10"

});

exports.handler = (event, context, callback) => {

const params = {

Key: {

id: {

S: event.id

}

},

TableName: process.env.TABLE\_NAME

};

dynamodb.getItem(params, (err, data) => {

if (err) {

console.log(err);

callback(err);

} else {

callback(null, {

id: data.Item.id.S,

title: data.Item.title.S,

watchHref: data.Item.watchHref.S,

authorId: data.Item.authorId.S,

length: data.Item.length.S,

category: data.Item.category.S

});

}

});

};

Save\_course

const AWS = require("aws-sdk");

const dynamodb = new AWS.DynamoDB({

region: process.env.AWS\_REGION,

apiVersion: "2012-08-10"

});

const replaceAll = (str, find, replace) => {

return str.replace(new RegExp(find, "g"), replace);

};

exports.handler = (event, context, callback) => {

const id = replaceAll(event.title, " ", "-").toLowerCase();

const params = {

Item: {

id: {

S: event.id

},

title: {

S: event.title

},

watchHref: {

S: `http://www.pluralsight.com/courses/${id}`

},

authorId: {

S: event.authoId

},

length: {

S: event.length

},

category: {

S: event.category

}

},

TableName: process.env.TABLE\_NAME

};

dynamodb.putItem(params, (err, data) => {

if (err) {

console.log(err);

callback(err);

} else {

callback(null, {

id: params.Item.id.S,

title: params.Item.title.S,

watchHref: params.Item.watchHref.S,

authorId: params.Item.authorId.S,

length: params.Item.length.S,

category: params.Item.category.S

});

}

}

)

}

update\_course.js

const AWS = require("aws-sdk");

const dynamodb = new AWS.DynamoDB({

region: "eu-central-1",

apiVersion: "2012-08-10"

});

exports.handler = (event, context, callback) => {

const params = {

Key: {

id: {

S: event.id

}

},

ExpressionAttributeNames: {

"#title": "title",

"#authorId": "authorId",

"#length": "length",

"#category": "category",

"#watchHref": "watchHref"

},

ExpressionAttributeValues: {

":title": {

S: event.title

},

":authorId": {

S: event.authorId

},

":length": {

S: event.length

},

":category": {

S: event.category

},

":watchHref": {

S: event.watchHref

}

},

UpdateExpression: "SET #title = :title, #watchHref = :watchHref, #authorId = :authorId, #length = :length, #category = :category",

TableName: process.env.TABLE\_NAME,

ReturnValues: "ALL\_NEW",

ConditionExpression: "attribute\_exists(id)"

};

dynamodb.updateItem(params, (err, data) => {

if (err) {

console.log(err);

callback(err);

} else {

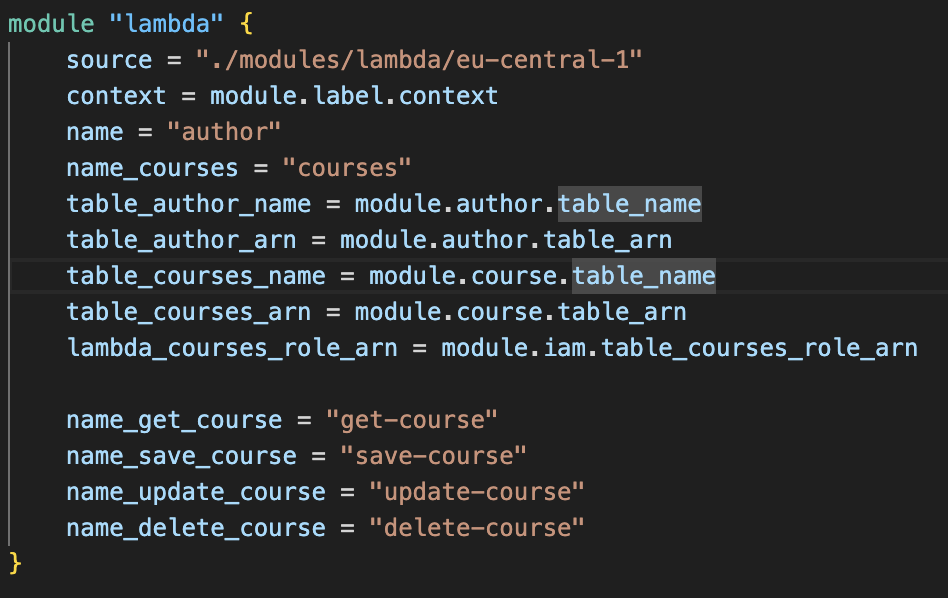
callback(null, data.Attributes);

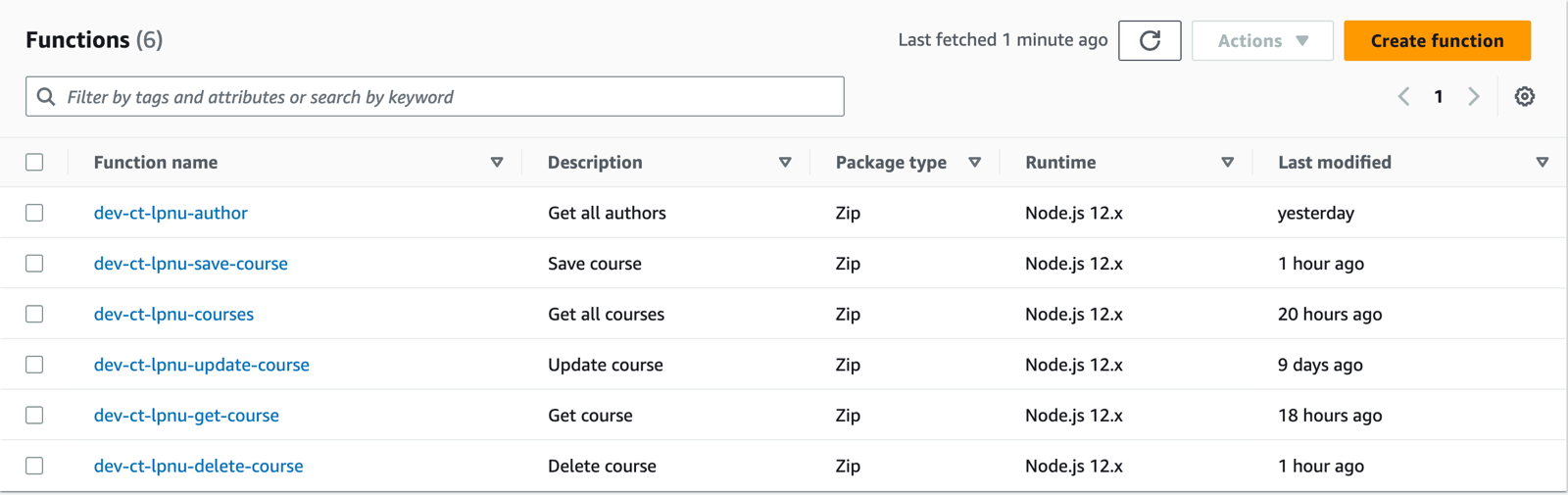
}

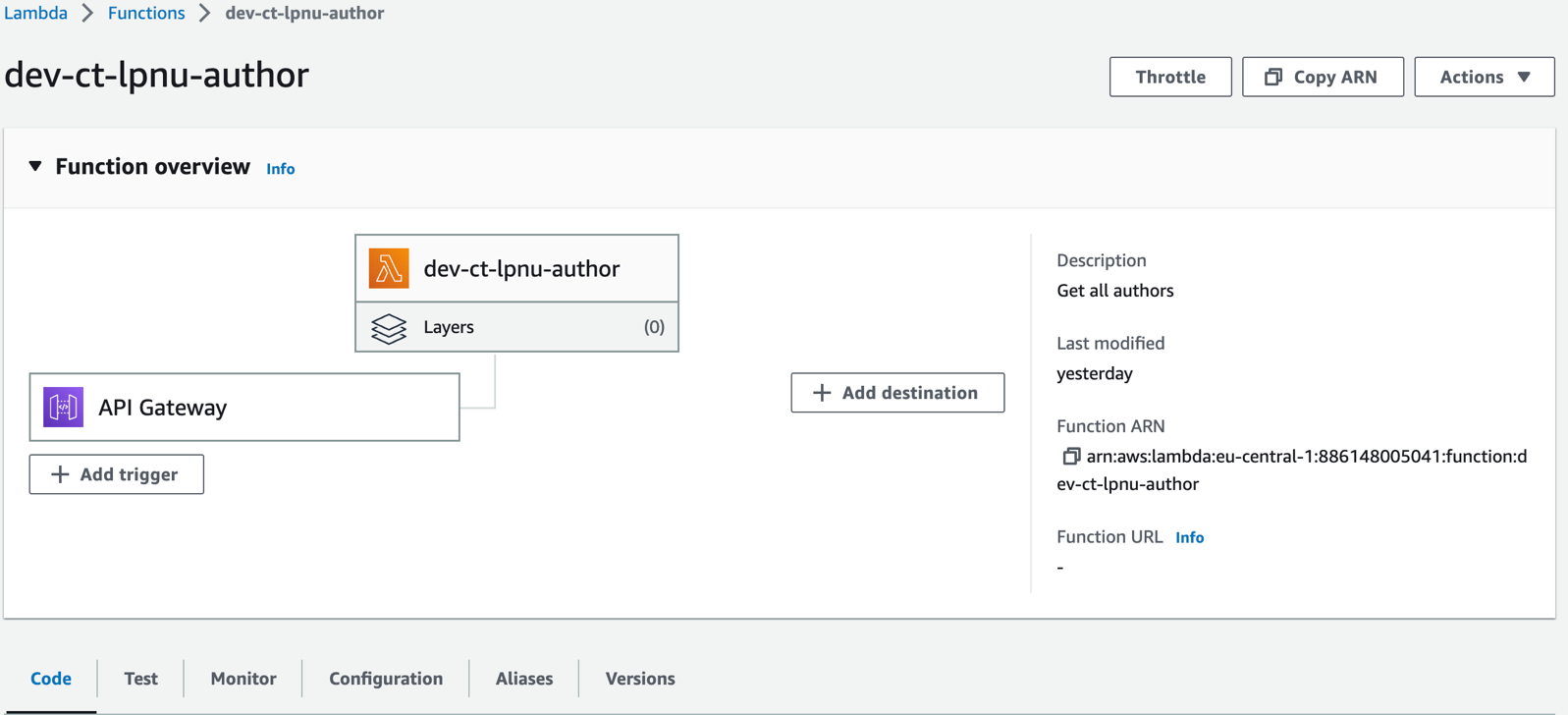
});

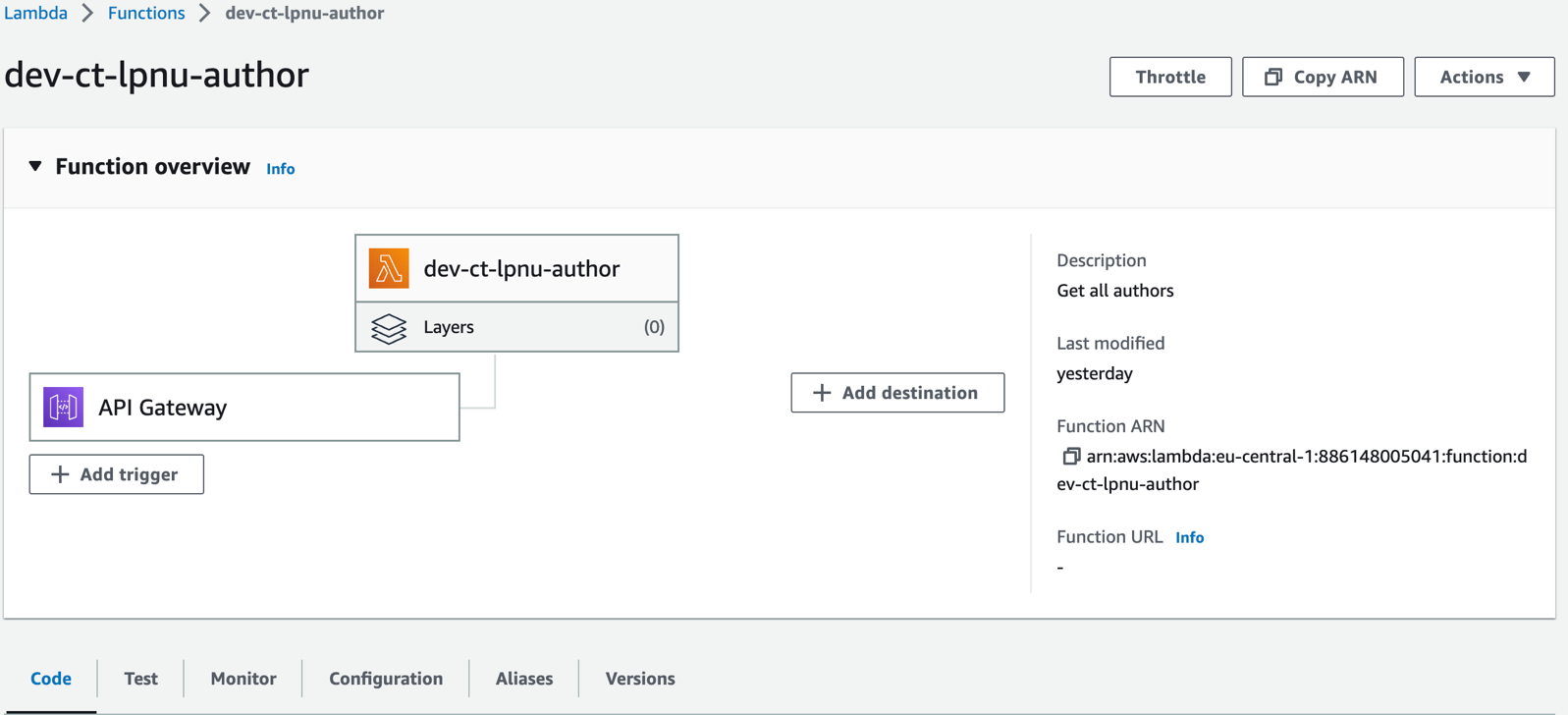
};

Для того, щоб ці lambda функції відображалися в aws в основному main.tf треба написати наступний код:



Після чого, на вкладці lambda function будуть усі створені функції:

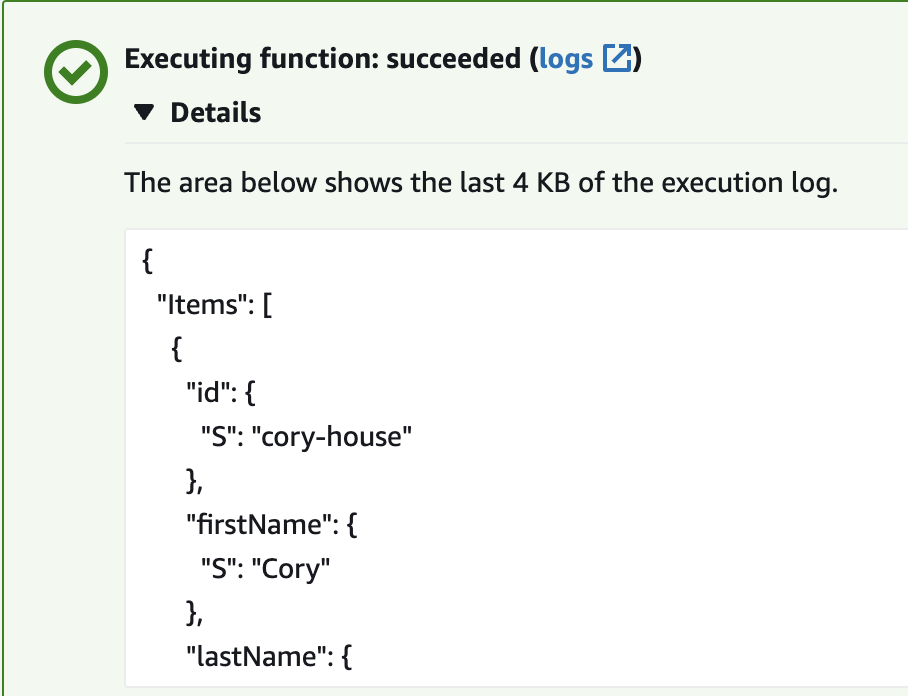
Для перевірки функції необхідно натиснути на функцію, після чого відкриється вкладка з основною інформацією про функцію:

У вкладці Test ми зможемо перевірити, чи функція працює, а також в залежності від того, який функціонал виконує функція, ми можемо модифікувати TestEvent:

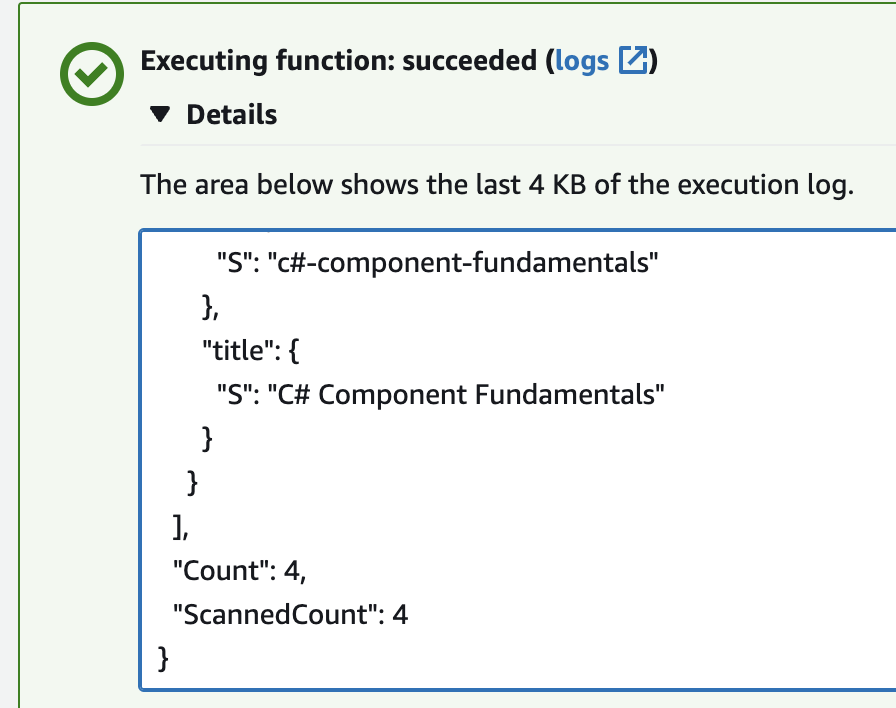
Натиснувши на кнопку Test ми розпочнемо тестування, і якщо все пройде успішно, то в залежності від функції ми отримаємо результати.

Виконаємо тестування створених раніше lambda функцій:

get-all-authors

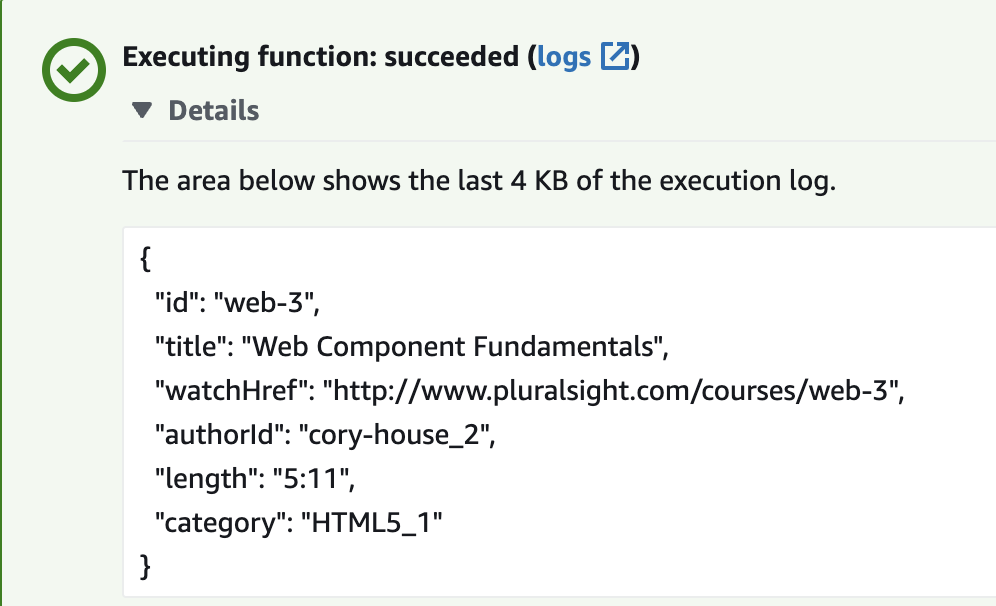


get-all-courses

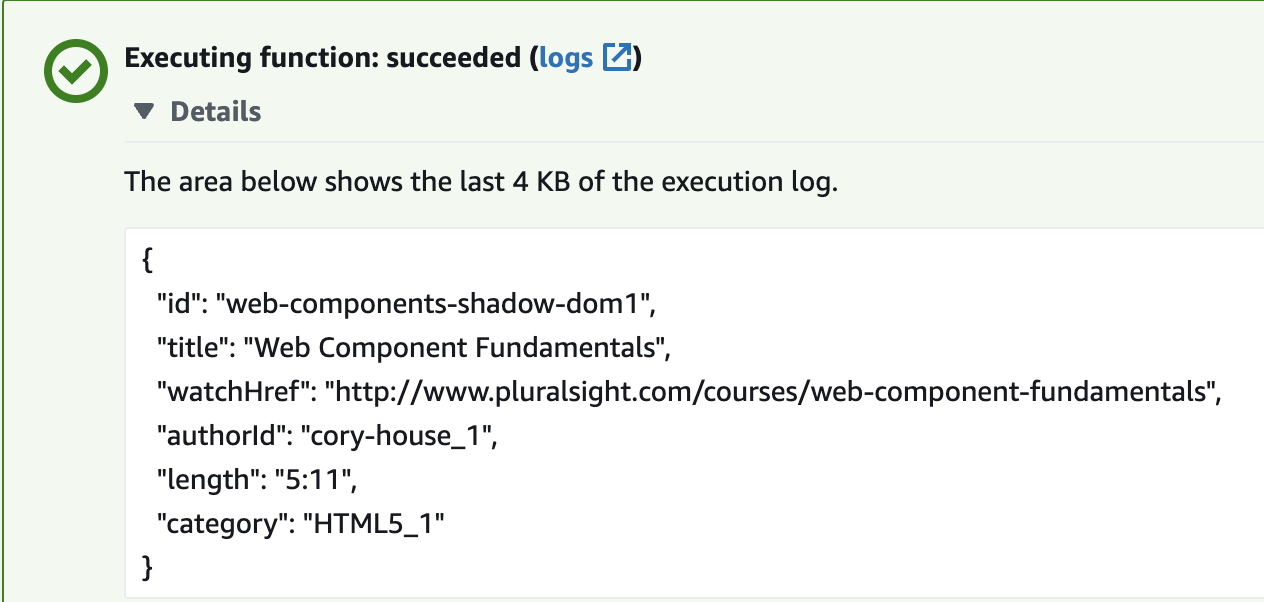


get-course

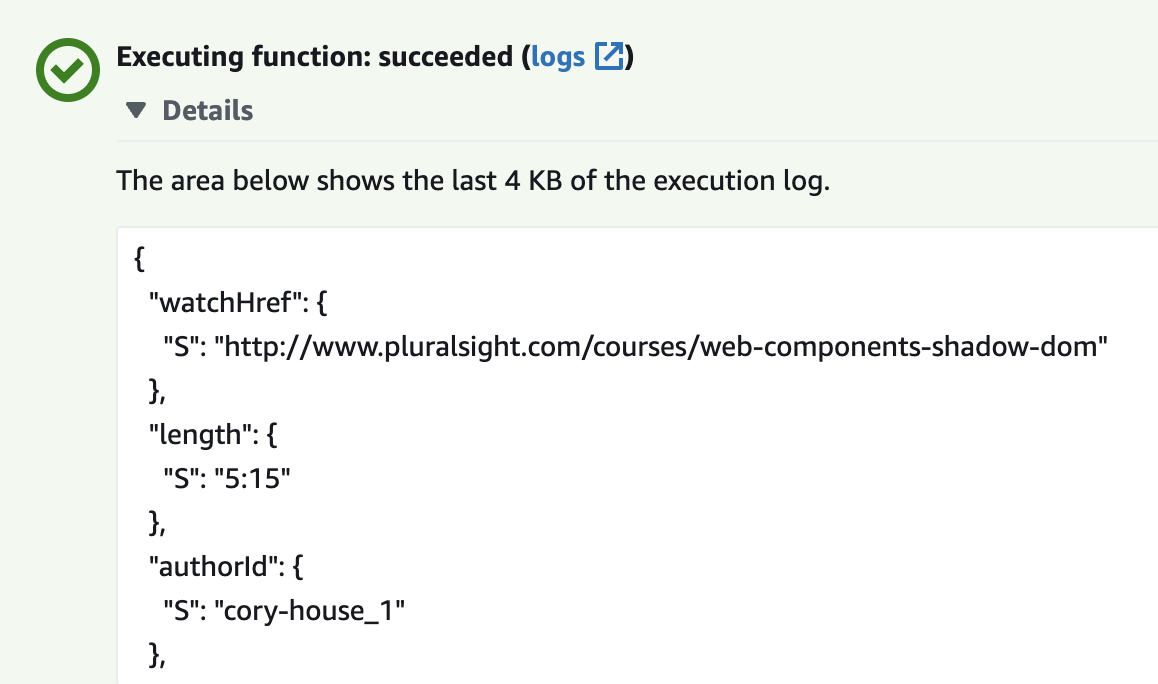
Для перевірки цієї функції необхідно в event JSON вписати Id потрібного запису



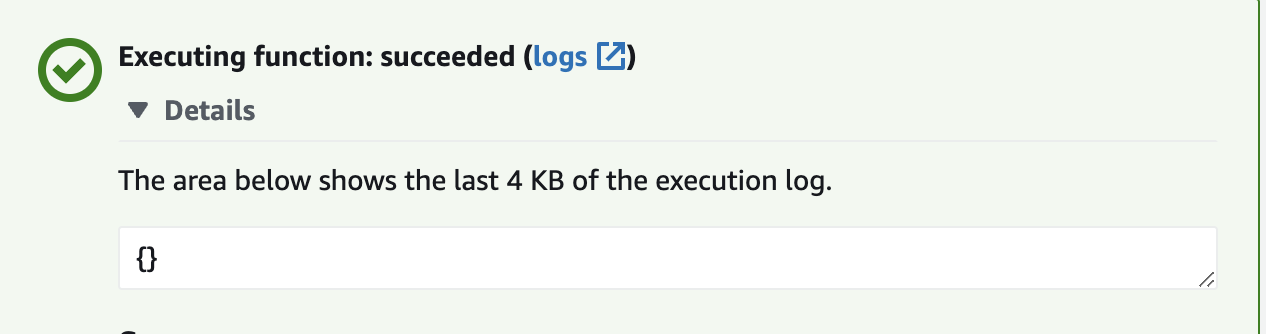
save-course



update-course



delete-course



**Висновок:** під час опрацювання лабораторної роботи я навчився зберігати дані у DynamoDB та створювати lambda функції