Naskah Soal

Lomba Kompetensi Siswa (LKS)

Tingkat Kabupaten Banjarnegara

Cloud Computing

2024

Project Description

Name : Rest API with lambda support

Congratulations!

Today you are working in consultation based company and your role as a ***Cloud Engineer***, at the first day you have responsibility to handle new project to deploy application. Your Solution Architect already have detail architecture based on discuss with client before.

This project is to build web based application with high-availability support, and there architecture required to support scalable, security, effectiveness, and manageable.

One of new technology will be used on this project is serverless, this new technology for event-driven compute services without provisioning or managing servers, it will be integrated with traditional infrastructure (EC2).

The goal of this project you need create CRUD application with ReactJS and will integrated to RestAPI using Amazon API Gateway, and last Lambda function for event-driven communication between API Gateway & Databases.

To ensure application running well & as expected, please follow all requirement & project details. You must get done this project by today.

Regards,
Your Solution Architect.

Summary of Tasks

1. Read the documentation thoroughly (Outlined below).
2. Please view and understand the architecture section.
3. Please carefully read the technical details section
4. Please carefully read the application details.
5. Log in to https://aws.amazon.com/console
6. Create your key pair you can use putty or generate with command, use this key pair to each instance you created & make sure you can SSH with this key. Upload your private key to https://s.id/lkscloud
7. Set up your VPC Configuration include VPC Endpoint, Internet gateway & more.
8. You can read VPC Configuration Details in the *Architecture Details section.*
9. Setup a security group, you can read security additional rules in *Security Service Details* section.
10. Setup databases, more details about it in *Details Service section.*
11. Create IAM & apply policy, you must follow instruction to ensure security is running properly.
12. Setup serverless application with AWS Lambda, the code of function already attached on this document.
13. Setup API Gateway services, Please read technical details for more information.
14. Make sure API Gateway & Lambda running well, it will be test with *curl* command.
15. Deploy web based application & will integrated with API Gateway.
16. Create Amazon Machine Images (AMI) for web based application.
17. Deploy & setup EC2 for support High Availability.
18. Verifying you can access application from public network (laptop) & make sureyour application is running well.

1. Introduction

This document contains the Low Level Design (LLD) for the project. This document is complemented by the system design document and other artifacts provided by judges.

The purpose of this document is to describe the project architecture in detail for the first infrastructure deployment. This includes overview of project architecture, components and technologies used to build the solution, including configuration details and application specific details.

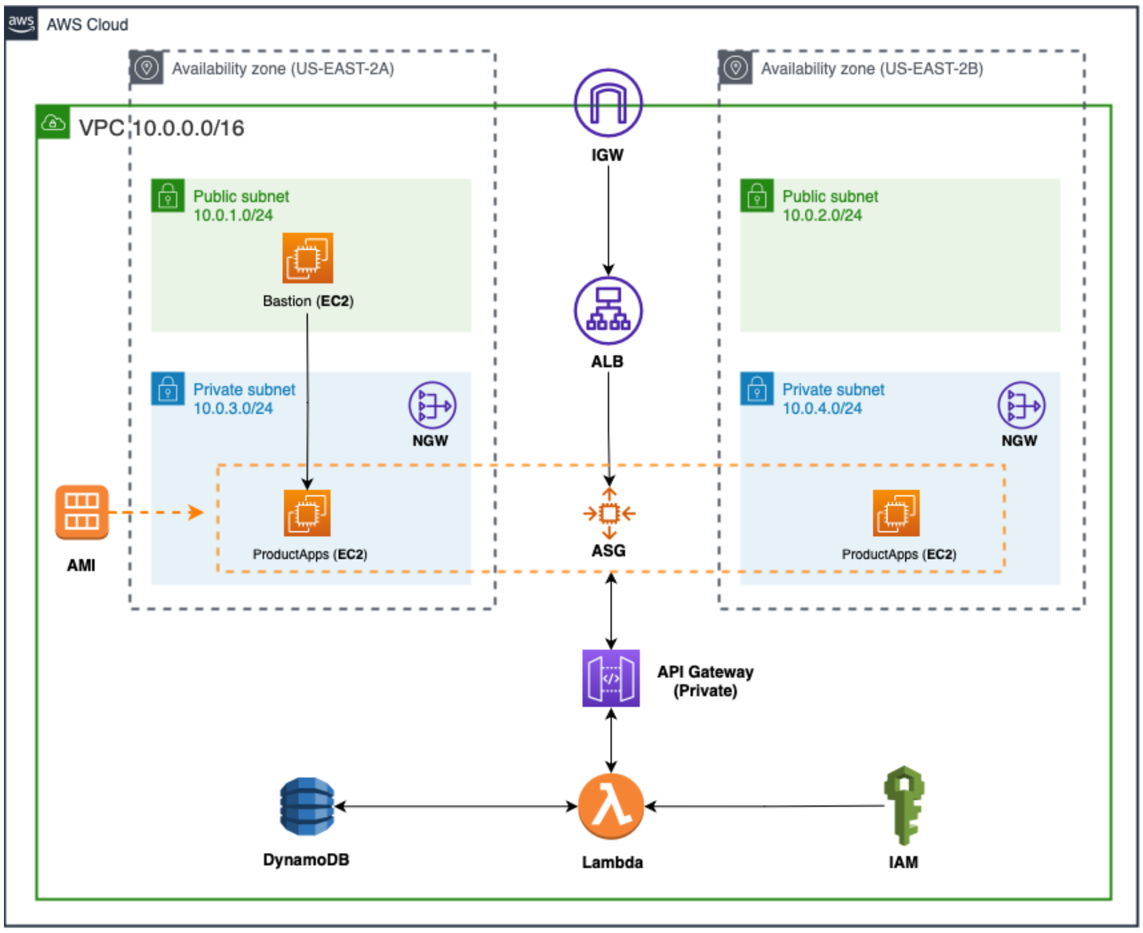
This document does not cover operational or change processes.

2. Design Overview

This section provides a brief but concrete overview of the design decisions taken in this project and key implementation details.

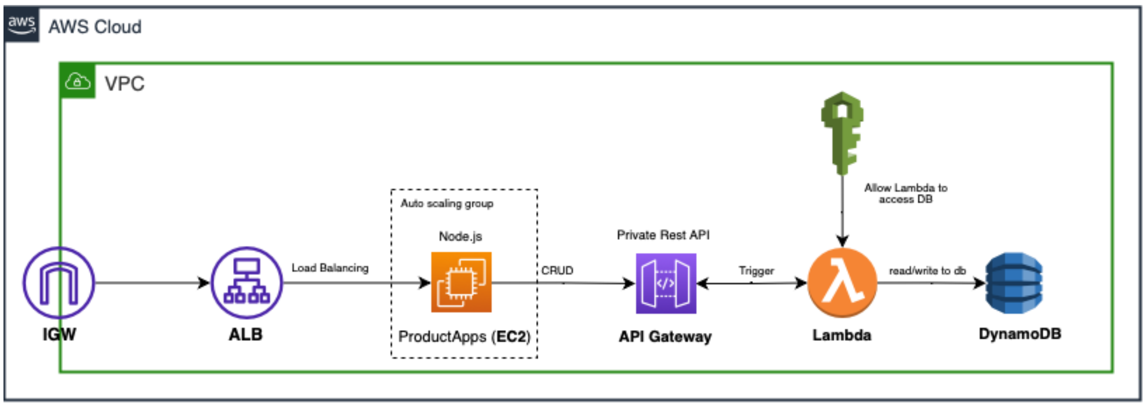
1. Compute;
    a. Instances with auto scaling support
    b. Image Templating with AMI
2. Security & Governance;
    a. AWS Identity & Access Management (IAM)
    b. Daw
3. Networking;
    a. Multi availability zone (AZ) will be used.
    b. Network segregation between public & private network for each services.
    c. Access from outside, will be used application load balancer & only HTTP/HTTPS port exposed.
    d. Private network will get internet connection via NAT Gateway.
4. Serverless;
    a. Lambda function for event-driven compute.
5. Databases;
    a. NoSQL database will used.
6. High-Availability;
    a. Application support scalable infrastructure for minimized down time.

3. Project Design


AWS Cloud
Availability zone (US-EAST-2A)
Availability zone (US-EAST-2B)
VPC 10.0.0.0/16
IGW
Public subnet 10.0.1.0/24
Bastion (EC2)
Public subnet 10.0.2.0/24
ALB
Private subnet 10.0.3.0/24
NGW
Private subnet 10.0.4.0/24
NGW
AMI
ProductApps (EC2)
ASG
ProductApps (EC2)
API Gateway (Private)
DynamoDB
Lambda
IAM

Picture 1. Project Infrastucture

4. Application Flow



Picture 2. Application flow

5. Network Configurations

5.1.1 Virtual Private Cloud (VPC)

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center.

5.1.2 Subnet & VPC

A subnet is a range of IP addresses in your VPC.

Services	Name	Subnet
VPC	vpc-lks	10.0.0.0/16
Subnet	public-2a	10.0.1.0/24
	public-2b	10.0.2.0/24
	private-2a	10.0.3.0/24
	private-2b	10.0.4.0/24

Please make sure during creating VPC enabled **DNS Hostname & DNS Resolution.**

5.1.3 Route Tables

A route table contains a set of rules, called routes, that are used to determine where network traffic from your subnet or gateway is directed. You must setting routing table for the environment public subnet and private subnet.

AZ	Name	Destination	Target	Subnet
US-EAST-2A	public-route-2a	VPC CIDR	Local	public-2a
		0.0.0.0/0	igw-lks	
	private-route-2a	VPC CIDR	Local	private-2a
		0.0.0.0/0	ngw-useast-2a	
US-EAST-2B	public-route-2b	VPC CIDR	Local	public-2b
		0.0.0.0/0	igw-lks	
	private-route-2b	VPC CIDR	Local	private-2b
		0.0.0.0/0	ngw-useast-2b	

5.1.4 Internet Gateway

An internet gateway is a horizontally scaled, redundant, and highly available VPC component that enables communication between your VPC and the internet.

Services	Name	VPC
Internet Gateway	igw-lks	vpc-lks

5.1.5 NAT gateway

Network Address Translation (NAT) gateway to enable instances in a private subnet to connect to services outside your VPC but prevent such external services from initiating a connection with those instances.

Name	For subnet	Type
ngw-useast-2a	private-2a	Public
ngw-useast-2b	private-2b	Public

5.1.6 Endpoint

A VPC endpoint enables connections between a virtual private cloud (VPC) and supported services, in this project used for connection between Private API Gateway & VPC. Please follow detail bellow :

- Name: product-api-endpoint
- Category : AWS Services
- Services : com.amazonaws.us-east-2.execute-api
- DNS Name : Enabled
- VPC : vpc-lks
- Subnet : All Subnet
- Security Groups : product-api-sg

6. Security & Identity Management

6.1 Key Pairs

A key pair, consisting of a public key and a private key, is a set of security credentials that you use to prove your identity when connecting to EC2 instances.

Add your keypair to AWS, it’s **mandatory** because will be used to access instances & install products application.

For this task, you can used **putty** for generate public & private key or using command bellow, make sure you created .pem.

```
ssh-keygen
```

6.2 AWS Identity and Access Management (IAM)

IAM is a web service for securely controlling access to AWS resources. It enables you to create and control services for user authentication or limit access to a certain set of people who use your AWS resources. This need ensure AWS Lambda & Databases can communicate, so create roles bellow :

Create new **Roles** with detail bellow :

Name	**Trusted Entity**	**Permission**
lambda-dynamodb-role	AWS Services, Lambda	CloudWatchFullAccess, AmazonDynamoDBFullAccess
products-lauch-role	AWS Services, EC2	AmazonSSMManagedInstanceCore

6.3 Security Groups

A security group acts as a virtual firewall that controls the traffic for one or more instances.

Name	**Description**	**Rule**	**Inbound**
product-api-sg	Allow VPC to Endpoint	HTTP, HTTPS	0.0.0.0/0
private-ec2-sg	Allow Bastion to Private	SSH, HTTP, HTTPS	0.0.0.0/0
bastion-sg	Allow Public to bastion	SSH	0.0.0.0/0

7. Surrounding apps & Infrastucture

7.1 Databases

Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. We used it with configuration bellow :

- *Table name: products*
- *Partition key: id*
- *Settings: Default settings*

7.2 AWS Lambda

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. This project used lambda for automate write & read data to DynamoDB (databases) triggered from AWS API Gateway call.

Function for this project :

- *Function: Author from scratch*
- *Function name: lambda-function-products*
- *Runtime: Node.js 14.x*
- *Role name:*
- *File name : index.js*

Please use this code to running a lambda function :

```
const AWS = require("aws-sdk");
const dynamo = new AWS.DynamoDB.DocumentClient();

exports.handler = async (event, context) => {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json"
  };

  try {

    console.info("event data: " + JSON.stringify(event))

    switch (event.httpMethod + " " + event.resource) {


      //Delete a single product by id
      case "DELETE /products/{id}":
        await dynamo
          .delete({
            TableName: "products",
            Key: {
              id: event.pathParameters.id
            }
          })
          .promise();
        body = `Deleted item ${event.pathParameters.id}`;
        break;

      //Get a single product by id
      case "GET /products/{id}":
        body = await dynamo
          .get({
            TableName: "products",
            Key: {
              id: event.pathParameters.id
            }
          })
          .promise();
        break;

      //Get all products in the table
      case "GET /products":
        body = await dynamo.scan({ TableName: "products" }).promise();
        break;

      //Put a single product in the table
      case "PUT /products":
        let requestJSON = JSON.parse(event.body);
        await dynamo
          .put({
            TableName: "products",
            Item: {
              id: requestJSON.id,
              price: requestJSON.price,
              name: requestJSON.name
            }
          })
          .promise();
        body = `Put item ${requestJSON.id}`;
        break;


      //If no route found output message with all even
      default:
        throw new Error(`Unsupported route: "${event.httpMethod + " " +
event.resource + " - EVENT: " + JSON.stringify(event)}"`);
```

```
    }
  } catch (err) {
    statusCode = 400;
    body = err.message;
  } finally {
    body = JSON.stringify(body);
  }

  return {
    statusCode,
    body,
    headers
  };
};
```

7.3 API Gateway

Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.

This project used **private** API gateway, only services from VPC can access a gateway.

Follow detail bellow :

- Protocol: REST API Private
- Create new API: New API
- API name: product-api
- Endpoint Type: Private
- Resource :
  - Resource name : products
  - Child Resource name: id
  - Child Resource path: {id}
- Enabled Use Lambda Proxy integration
- Method :
  - /products
    - GET
    - PUT
  - /products/id
    - GET
    - DELETE
- Integration Type : Lambda Function
- Lambda Name : *lambda-function-products*
- Deployment stage : dev
- API Gateway resource policy

API Gateway resource policy

Notes : changes VPC ID first

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                   "aws:SourceVpc": "vpc-xxxxxxxxxxx"
                }
            }
        }
    ]
}
```

8. Application Deployment

8.1 Bastion Host

- AMI : Ubuntu 22.04
- Security Groups : bastion-sg
- Auto Assign Public IP
- Subnet : public-2a

Copy private key from your laptop, then changes permission to 400, it will be used access from bastion to Private product VM with command :

```
nano <your-private-key-name>.

chmod 400 <your-private-key>
ssh ubuntu@<private-ip-vm-product> -i <your-private-key >
```

8.2 VM Product Apps (template)

Before create custom AMI, you need create based from bellow :

- AMI : Ubuntu 22.04
- Security Groups : private-ec2-sg
- Subnet : private-2a

8.3 React.Js apps

Inside of VM Products apps you need deploy product application & will be integrated with Private API Gateway, but not in this section because out of scope, today only for installation :

https://github.com/assyafii/product-apps

8.4 Product Apps AMI

Create AMI based on VM product apps

- AMI Name : product-apps-ami

9. Compute & High Availability

9.1 Launch Templates

Amazon Machine Image (AMI) was created from the products-apps instance. You must create a launch template by using this AMI.

- Name: products-launch
- AMI: products-apps-ami
- Instance Type: t2.micro
- Key Pair: Your key pair
- Security Group: private-ec2-sg
- IAM Instance Profile: products-lauch-role

9.2 Auto Scaling Group

Auto Scaling groups are collections of Amazon EC2 instances that enable automatic scaling and fleet management features.

- Name : product-asg
- Launch template: products launch
- VPC: vpc-lks
- Subnets: private-2a and private-2b
- Desired Capacity: 2
- Minimum Capacity: 2
- Maximum Capacity: 3
- Tracking Scaling Policy:
  - Metric Type: Average CPU Utilization
  - Target Value: 25
- Instances need: 60
- LB : Existing LB

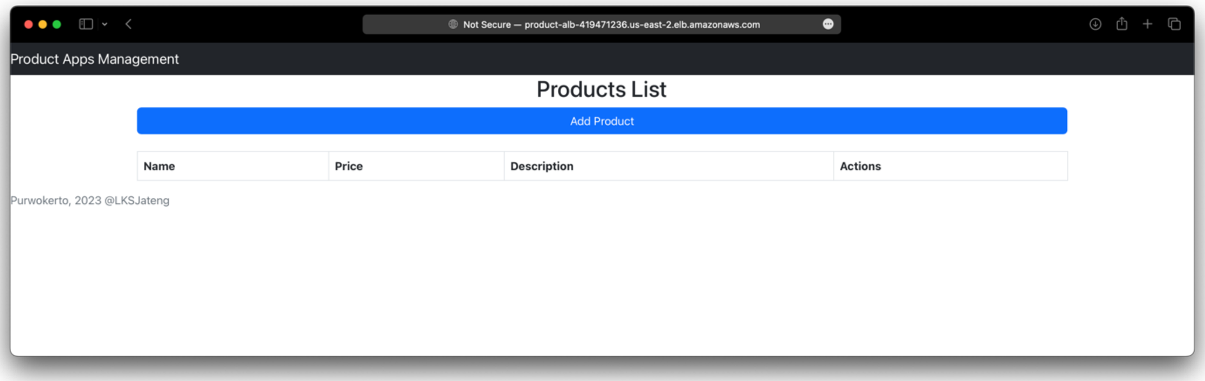
9.3 Application Load Balancer

Now that you web application server instances are deployed in private subnets, you need a way for the outside world to connect to them. In this task, you will create a load balancer to distribute traffic across your private instances.

- Type: Application Load Balancer
- Name : alb-product-apps
- VPC: vpc-lks
- Subnets: public-2a & public-2b
- Security Group: product-api-sg
- Target Group: Create a new target group (register with instances)

10. Project Verification

10.1 Application


Not Secure — product-alb-419471236.us-east-2.elb.amazonaws.com
Add Product

Product Apps Management

Products List

Name	Price	Description	Actions

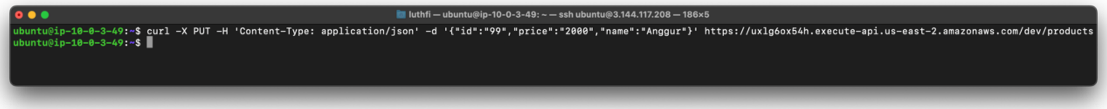
Purwokerto, 2023 @LKSJateng

10.2 CURL

- **PUT Data**

  Please add some data with curl from EC2 instances product (private-network), use bellow command :

```
curl -X PUT -H 'Content-Type: application/json' -d \
'{"id":"99","price":"2000","name":"Anggur"}' \
<your-api-gateway>/products
```

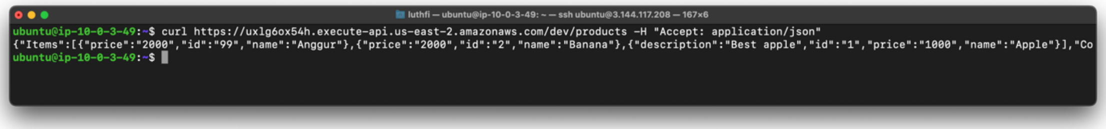

luthfi — ubuntu@ip-10-0-3-49: ~ — ssh ubuntu@3.144.117.208 — 186×5

```
ubuntu@ip-10-0-3-49:~$ curl -X PUT -H 'Content-Type: application/json' -d '{"id":"99","price":"2000","name":"Anggur"}' https://uxlg6ox54h.execute-api.us-east-2.amazonaws.com/dev/products
ubuntu@ip-10-0-3-49:~$
```

- **GET All Data**

  Same with before

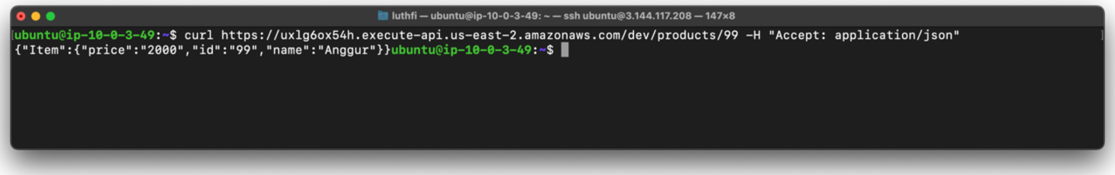
```
curl <your-api-gateway> -H "Accept: application/json"
```


luthfi — ubuntu@ip-10-0-3-49: ~ — ssh ubuntu@3.144.117.208 — 167×6

```
ubuntu@ip-10-0-3-49:~$ curl https://uxlg6ox54h.execute-api.us-east-2.amazonaws.com/dev/products -H "Accept: application/json"
{"Items":[{"price":"2000","id":"99","name":"Anggur"},{"price":"2000","id":"2","name":"Banana"},{"description":"Best apple","id":"1","price":"1000","name":"Apple"}],"Co
ubuntu@ip-10-0-3-49:~$
```

- **GET Specific Data**

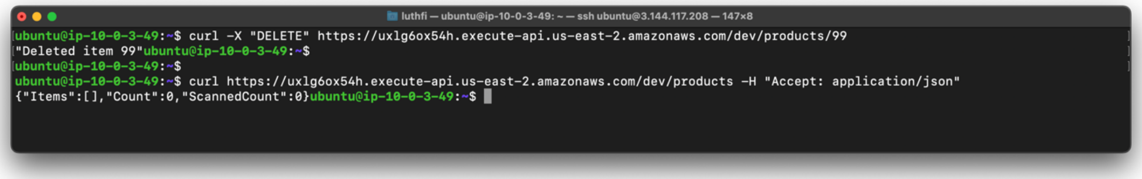
```
curl <your-api-gateway>/products/<id> -H "Accept: application/json"
```


luthfi — ubuntu@ip-10-0-3-49: ~ — ssh ubuntu@3.144.117.208 — 147×8

```
ubuntu@ip-10-0-3-49:~$ curl https://uxlg6ox54h.execute-api.us-east-2.amazonaws.com/dev/products/99 -H "Accept: application/json"
{"Item":{"price":"2000","id":"99","name":"Anggur"}}ubuntu@ip-10-0-3-49:~$
```

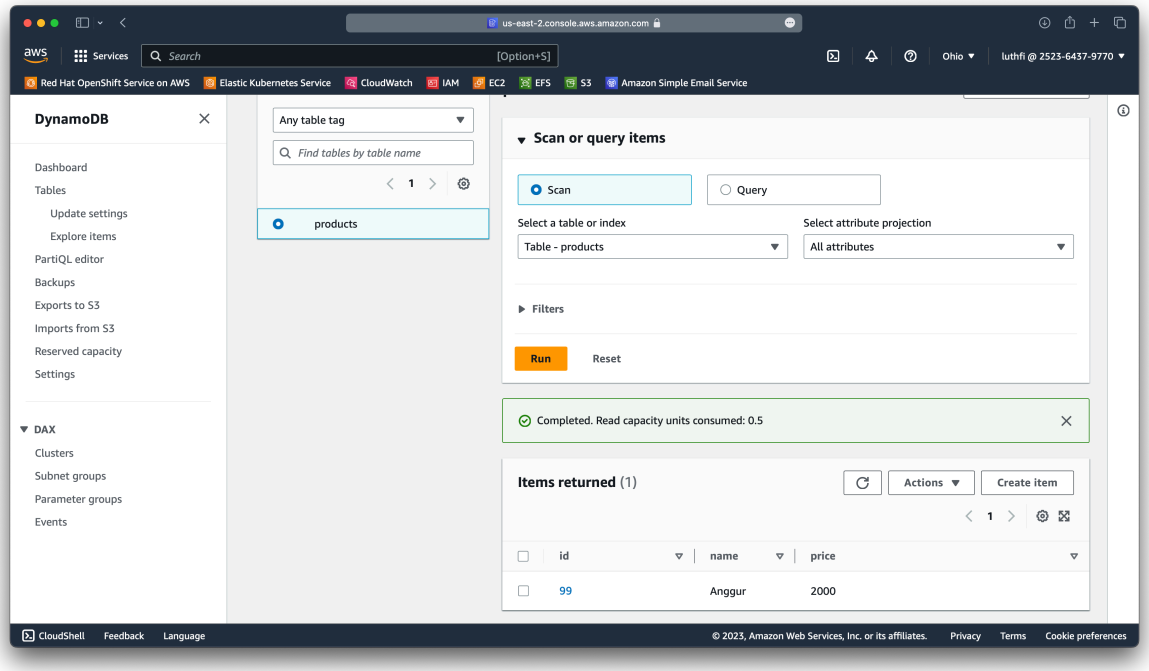
- **DELETE**

```
curl -X "DELETE" <your-api-gateway>/products/<id>
```


luthfi — ubuntu@ip-10-0-3-49: ~ — ssh ubuntu@3.144.117.208 — 147×8

```
ubuntu@ip-10-0-3-49:~$ curl -X "DELETE" https://uxlg6ox54h.execute-api.us-east-2.amazonaws.com/dev/products/99
"Deleted item 99"ubuntu@ip-10-0-3-49:~$
ubuntu@ip-10-0-3-49:~$
ubuntu@ip-10-0-3-49:~$ curl https://uxlg6ox54h.execute-api.us-east-2.amazonaws.com/dev/products -H "Accept: application/json"
{"Items":[],"Count":0,"ScannedCount":0}ubuntu@ip-10-0-3-49:~$
```

10.3 Databases



Select a table or index: Table - products

Select attribute projection: All attributes

Completed. Read capacity units consumed: 0.5

Items returned (1)

id	name	price
99	Anggur	2000