SolveIt

Submitted in partial fulfillment of the requirements of the degree of

BACHELOR OF COMPUTER ENGINEERING

By

Shreyash Divekar -

20102103

Rudra Chavan -

20102189

Department of Computer Engineering

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

(2022-2023)

CONTENTS

1. Introduction

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. Large clouds often have functions distributed over multiple locations, each of which is a data center. The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams.
Berkley simply defines Cloud Computing as ―Pay as you go SaaS because Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. The services themselves have long been referred to as Software as a Service (SaaS), so it is used. The datacenter hardware and software is what a Cloud is called. When a Cloud is made available in a pay-as-you-go manner to the public, it is called Public Cloud; the service being sold is Utility Computing. Private Cloud refers to internal datacenters of a business or other organization that are not made available to the public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not normally include Private Clouds.
Amazon Web Services, Inc. is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Oftentimes, clients will use this in combination with autoscaling. A cloud service has three distinct characteristics that differentiate it from traditional hosting. It is sold on demand, typically by the minute or the hour; it is elastic -- a user can have as much or as little of a service as they want at any given time; and the service is fully managed by the provider (the consumer needs nothing but a personal computer and Internet access). Significant innovations in virtualization and distributed computing, as well as improved access to high speed Internet and a weak economy, have accelerated interest in cloud computing.
In this project we have built a web application (SolveIt). SolveIt solves some basic mathematical calculations and responds with the solution in the form of an alert.
SolveIt is being deployed on AWS cloud platform with use of AWS Amplify web service. The process of smooth functioning of SolveIt involves various services provided by AWS.

2. Problem Statement and Objectives:

2.1 Problem Statement

The major focus of this project is on using a variety of AWS cloud services on the project that we built, the project can be deployed on AWS through multiple paths. The project can either be built on AWS platform itself or it can be uploaded in various ways like, upload the project through git providers or simply upload the zip file of the raw project that you created. Once the web application is deployed for smooth functioning of the app we use many cloud services provided by AWS, these cloud services serve the application in different phases of performing the designated task.

2.2 Objectives

The objectives of the project revolve around getting the web application running on AWS platform and increase the efficiency of web application.

- For deploying the web application we use AWS Amplify:

AWS Amplify is a set of purpose-built tools and features that enables frontend web and mobile developers to quickly and easily build full-stack applications on AWS. Amplify provides two services: Amplify Hosting and Amplify Studio. Amplify Hosting provides a git-based workflow for hosting full-stack serverless web apps with continuous deployment.

- Using lambda functions to perform operations and manage events:

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging.

- Use DynamoDB to store the solutions of the performed operations:

DynamoDB is a hosted NoSQL database offered by Amazon Web Services (AWS).

- Use API Gateway to send request and receive response between web app and database

3. Amazon Web Services:

Amazon provides a fantastic suite of web services that enables developers to create dynamic and robust applications. Deploying on AWS can save you time, money and manpower compared to building and maintaining more traditional systems. Amazon Web Services delivers a number of benefits for IT organizations and developers alike, including:

Cost-effective: Pay only for what you use, as you use it, with no up-front commitments. As the Amazon Web Services cloud grows, our operations, management and hardware costs shrink, and we pass the savings onto you.

Dependable: Utilize a battle-tested, web-scale infrastructure that handles whatever you throw at it. The Amazon Web Services cloud is distributed, secure and resilient, giving you reliability and massive scale.

Flexible: Build any application you want using any platform or any programming model. You control the resources you consume and fit them into your application as you see fit.

Comprehensive: Don‘t start from scratch. Amazon Web Services gives you a number of services you can incorporate into your applications. From databases to payments, these services help you build great applications cost effectively and with less up-front investment.

Various AWS cloud services that we use in this project are:

- Amplify hosting service
- Lambda function service
- DynamoDB
- API Gateway

4. Amplify Hosting service

Nowadays, most businesses are transitioning to a cloud-based model and are ready to cut the cord. Leading the Cloud - The Amazon Web Services (AWS) cloud platform is a leader in the various cloud adoption services.
Amazon's ever-growing portfolio offers a total of over 90 services and products to meet the needs of developers to develop fast, efficient, serverless, and secure web and mobile applications.
One such service that is gaining momentum and has greatly simplified and accelerated the developers' tasks is known as AWS Amplify. This article will allow a deeper dive into everything related to AWS Amplify - from how it works to its features, integrations, pricing model, upsides, and downsides.
What is AWS Amplify?
AWS is a full-suite platform developed to assist web and mobile developers in building full-stack and scalable applications powered by AWS. The platform comes with several tools and services that allow users to configure backends, connect apps, quickly deploy static web apps, and seamlessly manage content external to the AWS console.
AWS Amplify was launched in 2017 and is a full-suite package of tools and services specifically designed to help developers easily build and launch apps. The most important advantage of this tool is that it allows you to quickly and securely integrate myriad functions for everything from APIs to AI. It may also include code libraries, ready-to-use components, and built-in CLIs.
Another reason behind the launch of AWS Amplify is the user experience. User experience is the most integral component of any application that should be taken care of. AWS Amplify was built to integrate UX across multiple platforms, including web and mobile.
It gives users the flexibility to select the most comfortable building platform and is especially beneficial for frontend development. Most Amplify users also claim that it makes full-stack development more comfortable with its scalability factor.

How does AWS Amplify work?

You can think of AWS Amplify as a JavaScript library that lets you build and deploy serverless applications in the cloud. It is a full-stack application platform that combines both client-side and server-side code. In a nutshell, AWS Amplify consists of three major components:

- Libraries
- UI
- CLI Toolchain.

All these components work collaboratively to manage the application development lifecycle. Here's a brief look at each element:

- Libraries: This component lets you connect, integrate, and interact with AWS cloud services. The library makes it easy to add secure authentication, file storage, data storage, serverless APIs, analytics, push notifications, AR/VR, and many more features to your apps.
- UI: These are pre-built UI components designed around the cloud workflows in your application, including the authentication higher-order component.
- CLI Toolchain: This last component helps you scale your application. If you ever need to add more cloud services and features, easy-to-use CLI commands can efficiently make changes to your AWS-managed backend.

Advantages of Using AWS Amplify

1. Easy and UI Driven Development

AWS Amplify provides a simple, fast, and modern UI-driven approach to building mobile and web applications. The out-of-the-box UI component provides everything, so you don't have to code one. The design of CLI processes and workflows is also seamless, which speeds up app development.

2. Usage-based payments

Like many other paid AWS services, the payment model for AWS Amplify is very flexible and cost-effective as you only pay for the services you use.

3. Backend Support

AWS Amplify improves app performance by offering built-in support for the backend management.

4. It's free to get started

Several free and impressive tiers with AWS Amplify offer many benefits and zero cost. When you reach a higher threshold of technical requirements, you'll need to set up a paid tier.

5. Web-Based Analytics

AWS Amplify comes with a web-based analytics dashboard that is extremely useful for developers, designers, and project managers. It not only tracks user sessions and attributes but also provides in-app metrics. Analytics is always up to date, allowing teams to manage and track projects.

AWS Amplify Limitations

1. Higher Education Curve

If you are an AWS Amplify newbie, you can feel that the time you save in writing code will be used in learning the platform. Sometimes, it is difficult for beginners to find the right method in the documentation, and the multiple methods and versions sometimes make it difficult to navigate.

2. Constant Change

It is a constantly changing platform, and thus new changes and features are rolled out constantly. It means that an AWS Amplify user must always be on top to keep them up to date and continue exploring the platform.

3. Cost

Since AWS Amplify is a managed service, the end-user has less control over the environment and installed packages that can affect your website and most likely cost more than the backend can handle. It would be best to deal with the other pitfalls of managed services.

4. Traffic Distribution

You cannot use load balancers to distribute traffic when using AWS Amplify. It can be a major drawback in some cases, such as handling traffic spikes and latency issues.

5. AWS Lambda

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.

The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services.

The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so "serverless" doesn't mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of, so that you can focus on writing application code.

Each Lambda function runs in its own container. When a function is created, Lambda packages it into a new container and then executes that container on a multi-tenant cluster of machines managed by AWS. Before the functions start running, each function's container is allocated its necessary RAM and CPU capacity. Once the functions finish running, the RAM allocated at the beginning is multiplied by the amount of time the function spent running. The customers then get charged based on the allocated memory and the amount of run time the function took to complete.

The entire infrastructure layer of AWS Lambda is managed by AWS. Customers don't get much visibility into how the system operates, but they also don't need to worry about updating the underlying machines, avoiding network contention, and so on—AWS takes care of this itself.

And since the service is fully managed, using AWS Lambda can save you time on operational tasks. When there is no infrastructure to maintain, you can spend more time working on the application code—even though this also means you give up the flexibility of operating your own infrastructure.

Advantages of AWS Lambda:

AWS Lambda has a few unique advantages over maintaining your own servers in the cloud. The main ones are:

1. Pay per use.

In AWS Lambda, you pay only for the compute your functions use, plus any network traffic generated. For workloads that scale significantly according to time of day, this type of billing is generally more cost-effective.

2. Fully managed infrastructure

Now that your functions run on the managed AWS infrastructure, you don't need to think about the underlying servers—AWS takes care of this for you. This can result in significant savings on operational tasks such as upgrading the operating system or managing the network layer.

3. Automatic scaling.

AWS Lambda creates the instances of your function as they are requested. There is no pre-scaled pool, no scale levels to worry about, no settings to tune—and at the same time your functions are available whenever the load increases or decreases. You only pay for each function's run time.

4. Tight integration with other AWS products.

AWS Lambda integrates with services like DynamoDB, S3 and API Gateway, allowing you to build functionally complete applications within your Lambda functions.

Limitations of AWS Lambda:

1. Cold start time

When a function is started in response to an event, there may be a small amount of latency between the event and when the function runs. If your function hasn't been used in the last 15 minutes, the latency can be as high as 5-10 seconds, making it hard to rely on Lambda for latency-critical applications.

2. Execution time/run time.

A Lambda function will time out after running for 15 minutes. There is no way to change this limit. If running your function typically takes more than 15 minutes, AWS Lambda might not be a good solution for your task.

3. Memory available to the function.

The options for the amount of RAM available to the Lambda functions range from 128MB to 3,008MB with a 64MB step.

4. Code package size.

The zipped Lambda code package should not exceed 50MB in size, and the unzipped version shouldn't be larger than 250MB.

5. Concurrency.

By default, the concurrent execution for all AWS Lambda functions within a single AWS account are limited to 1,000. (You can request a limit increase for this number by contacting AWS support.)

6. DynamoDB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that require consistent single-digit millisecond latency at any scale. It is a fully managed database that supports both document and key-value data models. Its flexible data model and performance makes it a great fit for mobile, web, gaming, ad-tech, IOT, and many other applications. It is stored in SSD storage. It is spread across three geographically data centres.
Because of its availability in three geographically data centres, It consists of two different types of consistency models:

- Eventual Consistent Reads

It maintains consistency across all the copies of data which is usually reached within a second. If you read a data from DynamoDB table, then the response would not reflect the most recently completed write operation, and if you repeat to read the data after a short period, then the response would be the lattest update. This is the best model for Read performance.

- Strongly Consistent Reads

A strongly consistent read returns a result that reflects all writes that received a successful response prior to the read.
AWS DynamoDB Throughput Capacity:
DynamoDB throughput capacity depends on the read/write capacity modes for performing read/write operation on tables.
There are two types of read/write capacity modes:

- Provisioned mode

It defines the maximum amount of capacity that an application can use from a specified table.
In a provisioned mode, you need to specify the number of reads and writes per second required by the application.
If the limit of Provisioned mode throughput capacity is exceeded, then this leads to the request throttling.
A provisioned mode is good for applications that have predictable and consistent traffic.

The Provisioned mode consists of two capacity units:

1. Read Capacity Unit

The total number of read capacity units depends on the item size, and read consistency model.
Read Capacity unit represents two types of consistency models:
Strongly Consistent model: Read Capacity Unit represents one strong consistent read per second for an item up to 4KB in size.
Eventually Consistent model: Read Capacity Unit represents two eventually consistent reads per second for an item up to 4KB in size.
DynamoDB will require additional read capacity units when an item size is greater than 4KB. For example, if the size of an item is 8KB, 2 read capacity units are required for strongly consistent read while 1 read capacity unit is required for eventually consistent read.

2. Write Capacity Unit

The total number of write capacity unit depends on the item size.
Only 1 write capacity unit is required for an item up to size 1KB.
DynamoDB will require additional write capacity units when size is greater than 1KB. For example, if an item size is 2KB, two write capacity units are required to perform 1 write per second.
For example, if you create a table with 20 write capacity units, then you can perform 20 writes per second for an item up to 1KB in size.

- On-Demand mode

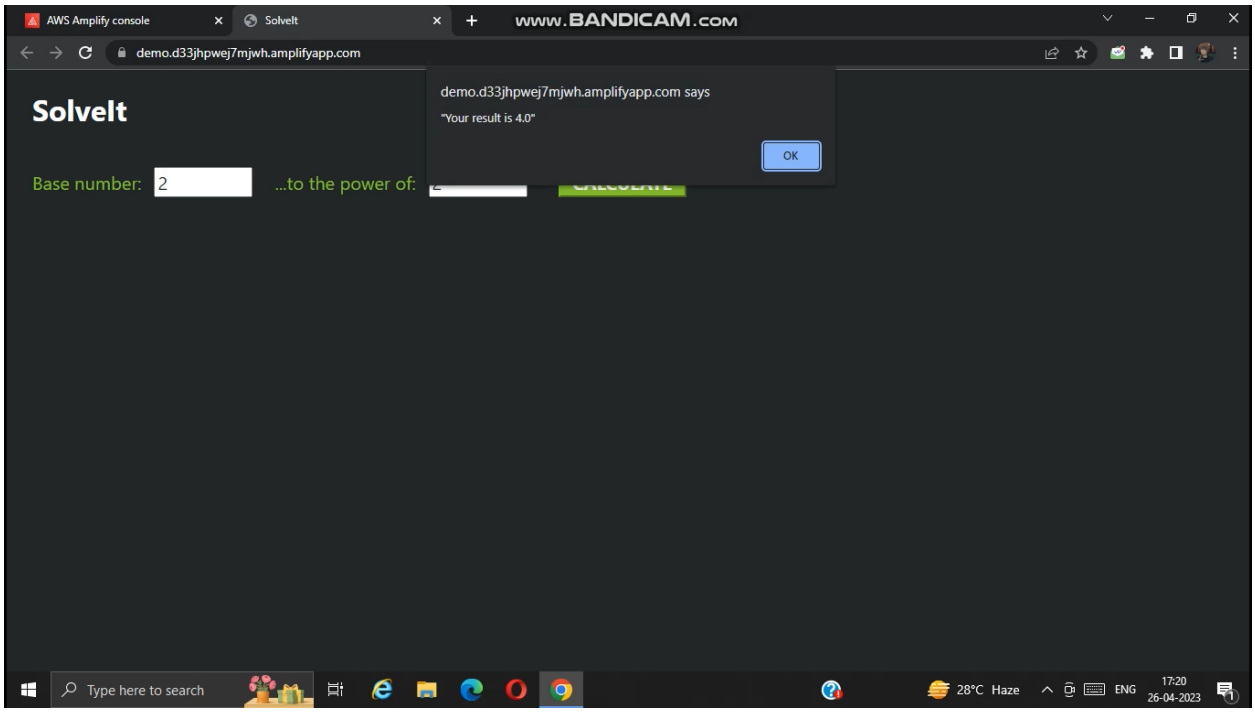
DynamoDB on-demand mode has a flexible new billing option which is capable of serving thousands of requests per second without any capacity planning.
On-Demand mode offers pay-per-request pricing for read and write requests so that you need to pay only for what you use, thus, making it easy to balance costs and performance. In On-Demand mode, DynamoDb accommodates the customer's workload instantly as the traffic level increases or decreases.
On-Demand mode supports all the DynamoDB features such as encryption, point-in-time recovery, etc except auto-scaling
If you do not perform any read/write, then you just need to pay for data storage only. On-Demand mode is useful for those applications that have unpredictable traffic and database is very complex to forecast.

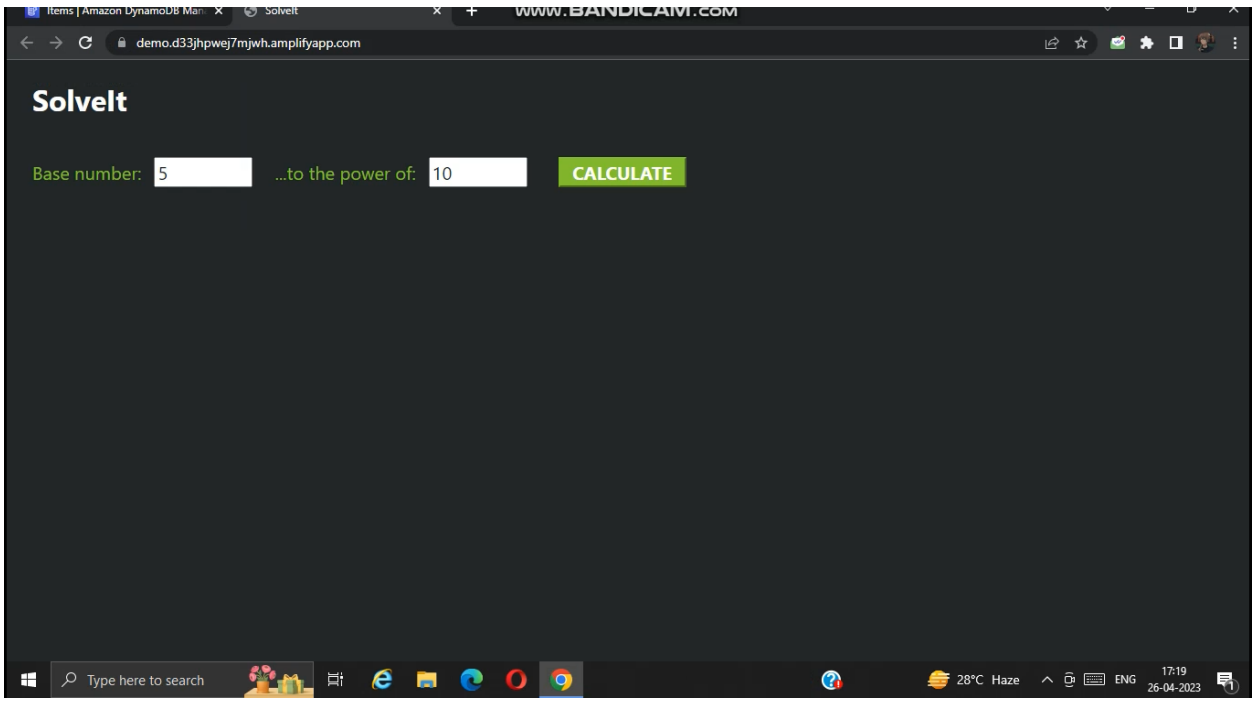
7. API Gateway

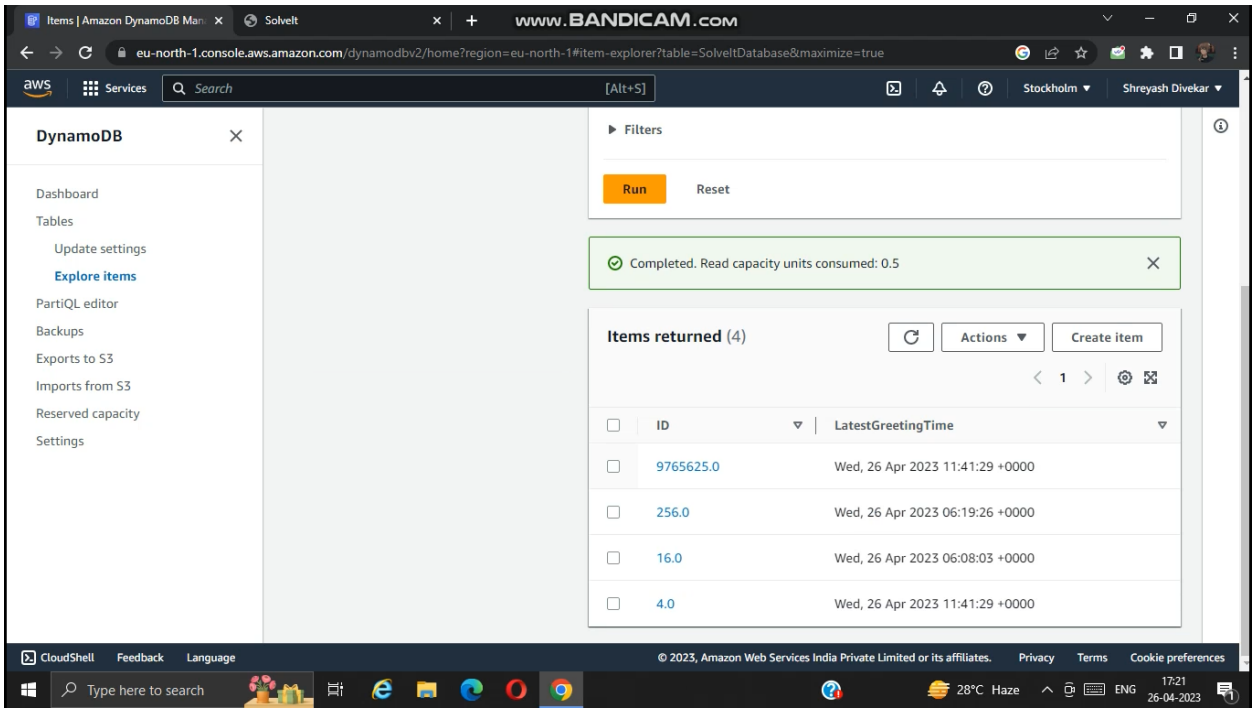
An API Gateway can handle any type of interaction between your website, web or mobile application or even IoT devices and your microservices.
It's a fully managed service that provides all the necessary tools for developers in order to create, publish, manage and secure your API regarding of scale. Amazon API Gateway will take care of all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management.
Deployable with a couple of clicks, your Amazon API Gateway will act as the single point of entry to your backend services and will handle the data management, business logic or any other type of functionality or workloads running on services like EC2, AWS Lambda and many more.
Written in Node.js, it can be configured as the backbone of the AWS Cloud as it is the gateway between all the connected services in the AWS ecosystem.
Amazon API Gateway is a closed-source software-as-a-service (SaaS) product written in Node.js available only on AWS. Amazon API Gateway can be considered a backplane in the AWS ecosystem.
In conjunction with AWS Lambda, the API gateway forms the client-facing part of Amazon's serverless infrastructure. Lambda runs the code on the highly available, fully managed computing infrastructure but relies on API gateway to expose those endpoints to the required services.
To enable the serverless applications, API Gateway supports streamlined proxy integrations with AWS Lambda and HTTP endpoints.
Features include:

- Build, deploy and manage APIs
- Resiliency
- API lifecycle management
- SDK generation
- API operations monitoring
- AWS authorization
- API keys for third-party developers

8. Implementation






Items | Amazon DynamoDB Ma
Solvelt
www.BANDICAM.com
eu-north-1.console.aws.amazon.com/dynamodbv2/home?region=eu-north-1#item-explorer?table=SolveltDatabase&maximize=true
aws
Services
Search
[Alt+S]
Stockholm
Shreyash Divekar
DynamoDB
Dashboard
Tables
Update settings
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Reserved capacity
Settings
Filters
Run
Reset
Completed. Read capacity units consumed: 0.5
Items returned (4)
Actions
Create item
1
ID
LatestGreetingTime
9765625.0
Wed, 26 Apr 2023 11:41:29 +0000
256.0
Wed, 26 Apr 2023 06:19:26 +0000
16.0
Wed, 26 Apr 2023 06:08:03 +0000
4.0
Wed, 26 Apr 2023 11:41:29 +0000
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
28°C Haze
ENG
17:21
26-04-2023

9. Conclusion

Thus we have deployed our simple responsive web app on AWS Cloud and used a variety of AWS cloud services. AWS has a very user friendly interface and provides a wide range of functionalities for the developer to explore. Many businesses can use AWS services to expand their service to clients, AWS benefits the other services providers as well by assisting them in setting up their services providing schemas. AWS not only works in storage and compute sector but also in security sector where it enables the user/developer to manage roles and policies.