Introduction

Amazon DynamoDB is a fully managed NoSQL database service provided by AWS. In this tutorial, we'll utilize Terraform to provision a DynamoDB table and populate it with sample data. By the end, you'll have a DynamoDB table ready for use in your applications.

Prerequisites:

Before we begin, make sure you have:
An AWS account with appropriate permissions
Terraform installed on your local machine
Basic understanding of AWS services and Terraform concepts

Step 1: Provider Configuration

Let's start by configuring the AWS provider in Terraform. Create a file named `main.tf` and add the following code:

```
provider "aws" {
region = var.region
access
_
key = var.access
_
secret
_
key = var.secret
_
key
key
}
```

This block configures the AWS provider with the specified region and authentication credentials using variables.

Step 2: Creating DynamoDB Table

Now, let's define our DynamoDB table:

```
resource "aws
_
dynamodb
_
table" "dynamodb
_
name = "test-table"
billing_
mode = "PROVISIONED"
read
_
capacity = 5
write
_
capacity = 5
table" {
hash
_
key = "RollNo.
"
https://medium.com/@nagarjun
_
nagesh/terraform-managing-data-with-aws-dynamodb-51a024ad92cd 1/311/1/25, 9:11 AM attribute {
name = "RollNo.
type = "N"
}
Terraform: Managing Data with AWS DynamoDB | by Nagarjun (Arjun) Nagesh | Medium
"
}
```

This resource creates a DynamoDB table named "test-table"
mode and a numeric hash key named "RollNo.
"
with provisioned billing

Step 3: Populating Table with Sample Data

Let’s add some sample items to our DynamoDB table:

```
resource "aws
_
dynamodb
table
item" "item1" {
_
_
table
name = aws
_
_
dynamodb
_
table.dynamodb
_
hash
_
key = aws
_
dynamodb
_
table.dynamodb
table.name
table.hash
_
_
item = <<ITEM
key
{
"RollNo.
": {"N": "1"},
"Name": {"S": "Arjun"}
}
ITEM
}
resource "aws
_
dynamodb
table
item" "item2" {
_
_
table
name = aws
_
_
dynamodb
_
table.dynamodb
_
hash
_
key = aws
_
dynamodb
_
table.dynamodb
table.name
table.hash
_
_
item = <<ITEM
key
{
"RollNo.
": {"N": "2"},
"Name": {"S": "Nagesh"}
}
ITEM
}
```

```
resource "aws
_
dynamodb
table
item" "item3" {
_
_
table
name = aws
_
_
dynamodb
_
table.dynamodb
_
hash
_
key = aws
_
dynamodb
_
table.dynamodb
table.name
table.hash
_
_
item = <<ITEM
key
{
"RollNo.
": {"N": "3"},
"Name": {"S": "Nagarjun"}
}
ITEM
}
```

These resources add sample items to the DynamoDB table with their respective RollNo. and Name attributes.

Step 4: Output

Let’s define an output to retrieve the ARN of the DynamoDB table:

```
output "table
arn" {
_
value = aws
_
dynamodb
_
table.dynamodb
table.arn
_
description = "DynamoDB Table created successfully"
}
```

This output provides the ARN of the DynamoDB table for reference.

Conclusion:

ve successfully provisioned a DynamoDB table and populated it with sample data using Terraform. DynamoDB offers scalable and fully managed NoSQL database capabilities, making it an excellent choice for various use cases, including web applications, gaming, IoT, and more. Explore further to customize and manage your DynamoDB tables according to your application requirements.

Happy data management!