BTS – MBDS
Big Data Infrastructure

A3: AWS - DynamoDB

Ennio Maldonado

03 March, 2021

DynamoDB – Example Table

Terraform script

main.tf

```
resource "aws_dynamodb_table" "basic-dynamodb-table" {
  name           = "GameScores"
  billing_mode   = "PROVISIONED"
  read_capacity  = 20
  write_capacity = 20
  hash_key       = "UserId"
  range_key      = "GameTitle"

  attribute {
    name = "UserId"
    type = "S"
  }

  attribute {
    name = "GameTitle"
    type = "S"
  }

  attribute {
    name = "TopScore"
    type = "N"
  }

  ttl {
    attribute_name = "TimeToExist"
    enabled        = false
  }

  global_secondary_index {
    name               = "GameTitleIndex"
    hash_key           = "GameTitle"
    range_key          = "TopScore"
    write_capacity     = 10
    read_capacity      = 10
    projection_type    = "INCLUDE"
    non_key_attributes = ["UserId"]
  }

  tags = {
    Name        = "dynamodb-table-1"
    Environment = "production"
  }
}
```

Terraform Apply –––>

```
❯ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_dynamodb_table.basic-dynamodb-table will be created
  + resource "aws_dynamodb_table" "basic-dynamodb-table" {
      + arn              = (known after apply)
      + billing_mode     = "PROVISIONED"
      + hash_key         = "UserId"
      + id               = (known after apply)
      + name             = "GameScores"
      + range_key        = "GameTitle"
      + read_capacity    = 20
      + stream_arn       = (known after apply)
      + stream_label     = (known after apply)
      + stream_view_type = (known after apply)
      + tags             = {
          + "Environment" = "production"
          + "Name"        = "dynamodb-table-1"
        }
      + write_capacity   = 20

      + attribute {
          + name = "GameTitle"
          + type = "S"
        }
      + attribute {
          + name = "TopScore"
          + type = "N"
        }
      + attribute {
          + name = "UserId"
          + type = "S"
        }

      + global_secondary_index {
          + hash_key           = "GameTitle"
          + name               = "GameTitleIndex"
          + non_key_attributes = [
              + "UserId",
            ]
          + projection_type    = "INCLUDE"
          + range_key          = "TopScore"
          + read_capacity      = 10
          + write_capacity     = 10
        }

      + point_in_time_recovery {
          + enabled = (known after apply)
        }

      + server_side_encryption {
          + enabled     = (known after apply)
          + kms_key_arn = (known after apply)
        }

      + ttl {
          + attribute_name = "TimeToExist"
          + enabled        = false
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
```

Terraform Init

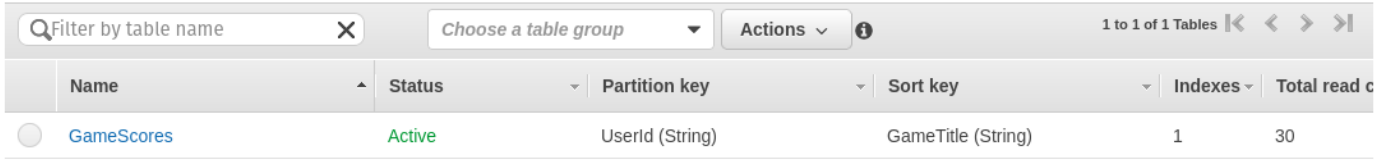
```
❯ terraform init
Initializing modules...
- disabled_dynamodb_table in ../..
- dynamodb_table in ../..

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/random...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/random v3.1.0...
- Installed hashicorp/random v3.1.0 (signed by HashiCorp)
- Installing hashicorp/aws v3.30.0...
- Installed hashicorp/aws v3.30.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

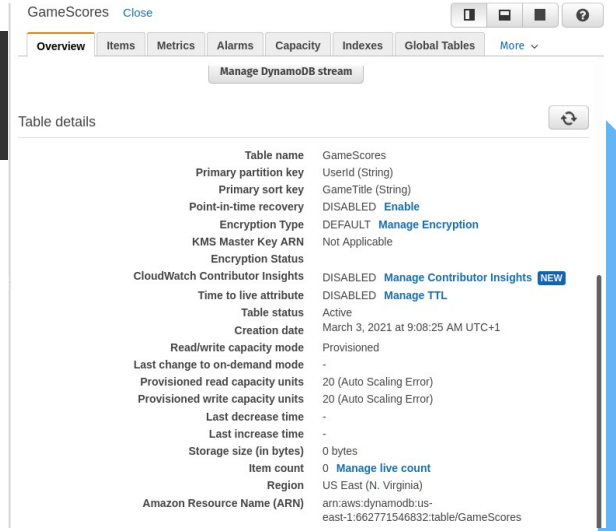
DynamoDB – Example Table

Terraform Output:

```
aws_dynamodb_table.basic-dynamodb-table: Creating...
aws_dynamodb_table.basic-dynamodb-table: Still creating... [10s elapsed]
aws_dynamodb_table.basic-dynamodb-table: Creation complete after 18s [id=GameScores]
```

Confirmed creation on DynamoDB dashboard





Added a record to the table

```
❯ aws dynamodb put-item --table-name GameScores --item '{ "UserId": {"S": "BTS-007"},  "GameTitle": {"S":
 "Fortnite"} ,  "TopScore": {"N": "100"} }' --return-consumed-capacity TOTAL
```

DynamoDB – Querying

```
❯ aws dynamodb scan --table-name GameScores
{
    "Items": [
        {
            "UserId": {
                "S": "BTS-007"
            },
            "GameTitle": {
                "S": "Fortnite"
            },
            "TopScore": {
                "N": "100"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
```

DynamoDB – Deleting resources

```
❯ terraform destroy

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_dynamodb_table.basic-dynamodb-table will be destroyed
  - resource "aws_dynamodb_table" "basic-dynamodb-table" {
      - arn            = "arn:aws:dynamodb:us-east-1:662771546832:table/GameScores" -> null
      - billing_mode   = "PROVISIONED" -> null
      - hash_key       = "UserId" -> null
      - id             = "GameScores" -> null
      - name           = "GameScores" -> null
      - range_key      = "GameTitle" -> null
      - read_capacity  = 20 -> null
      - stream_enabled = false -> null
      - tags           = {
```

MongoDB on DataBricks

Connect MongoDB to DataBricks followed this guide:

https://docs.databricks.com/data/data-sources/mongodb.html