DynamoDB Auto Scaling Management – Automation Script

Overview

This document describes the process of managing DynamoDB auto scaling using an automated Python script integrated into an Azure DevOps pipeline.

Purpose

To enable teams to dynamically manage DynamoDB billing mode and auto scaling configurations across environments using an automated and repeatable solution.

Problem Statement

Manually switching DynamoDB billing modes and managing auto scaling policies is error-prone and non-scalable. We needed a way to:

- Transition tables between `PROVISIONED` and `PAY_PER_REQUEST` modes.
- Register/Deregister auto scaling targets for read and write capacity.
- Automate this logic as part of our CI/CD pipeline.

Solution Summary

We implemented a Python script that performs the following:

- Accepts table name, region, billing mode, and scaling thresholds as input arguments.
- Switches billing mode from `PROVISIONED` to `PAY_PER_REQUEST` and vice versa.
- Deregisters or registers scalable targets for Read/Write Capacity Units accordingly.
- Integrates into Azure DevOps using a command-line pipeline task.

Script Arguments

```
--tableName      # Required: DynamoDB table name
--region         # Required: AWS region
--readMode       # Required: 'ondemand' or 'provisioned'
--writeMode      # Required: 'ondemand' or 'provisioned'
--minRead        # Optional: Minimum read capacity
--maxRead        # Optional: Maximum read capacity
--targetRead     # Optional: Target read utilization
--minWrite       # Optional: Minimum write capacity
--maxWrite       # Optional: Maximum write capacity
--targetWrite    # Optional: Target write utilization
```

Sample Python Logic (Key Excerpt)

```
parser = argparse.ArgumentParser()
parser.add_argument("--tableName", required=True, type=lambda s: s.strip())
parser.add_argument("--region", required=True)
parser.add_argument("--readMode", choices=["ondemand", "provisioned"],
required=True)
parser.add_argument("--writeMode", choices=["ondemand", "provisioned"],
required=True)
# Other args omitted for brevity
args = parser.parse_args()
```

Azure DevOps Pipeline Integration

```
- script: |
    cd others/dynamodb-scale
    source dynamoscaleenv/bin/activate
    python3 manage_dynamodb_autoscaling.py \
        --tableName "${{ parameters.tableName }}" \
        --region "${{ parameters.region }}" \
        --readMode "${{ parameters.readMode }}" \
        --writeMode "${{ parameters.writeMode }}" \
        --minRead "${{ parameters.minRead }}" \
        --maxRead "${{ parameters.maxRead }}" \
        --targetRead "${{ parameters.targetRead }}" \
        --minWrite "${{ parameters.minWrite }}" \
        --maxWrite "${{ parameters.maxWrite }}" \
        --targetWrite "${{ parameters.targetWrite }}"
  displayName: 'Configure DynamoDB Auto Scaling'
```

Common Errors and Their Meanings

Error	Meaning	Solution
`ObjectNotFoundException`	No scalable target found	Safe to ignore if the table was already on-demand or auto scaling was never set
`command not found`	Bad argument syntax in shell	Ensure arguments are passed after the script call and not interpreted as shell commands
Exit Code `127`	Command not found in script	Possibly a missing Python environment or syntax error in bash

Final Remarks

The implementation ensures safe, idempotent updates to DynamoDB billing and scaling configurations. Error logs and warnings are handled gracefully, making the pipeline resilient to unintended states.

This automation has reduced manual overhead, increased consistency, and can be extended to multiple tables and environments as needed.

Author: Sumit Kumar Singh\ **Last Updated:** June 24, 2025