# Dynamic DynamoDB Documentation

***Release 2.5.1***

**Sebastian Dahlgren**

**Aug 12, 2017**

# Contents

CHAPTER 1

# Using `pip`

Installing Dynamic DynamoDB is pretty straight forward using `pip`.

```
pip install dynamic-dynamodb
```

CHAPTER 2

# Manually via Git

It is also possible to install Dynamic DynamoDB manually, if you do no have access to `pip`. First checkout the project from GitHub:

```
git clone https://github.com/sebdah/dynamic-dynamodb
```

Then install the `dynamic-dynamodb` command locally.

```
cd dynamic-dynamodb
make install
```

# CHAPTER 3

# Using the CloudFormation template

Dynamic DynamoDB provides a CloudFormation template to make it easy to get started. You can read more about that in the cloudformation-template.

CHAPTER 4

# Configuration options

The sections below describe the various options available in the Dynamic DynamoDB configuration file. See _example_configuration for an example configuration file.

## Global configuration

**Section name:** `[global]`

| Option | Type | Default | Comment |
|---|---|---|---|
| aws-access-key-id | `str` | | AWS access API key |
| aws-secret-access-key-id | `str` | | AWS secret API key |
| check-interval | `int` | 300 | How many seconds to wait between the checks |
| circuit-breaker-timeout | `float` | 10000.00 | Timeout for the circuit breaker, in ms |
| circuit-breaker-url | `str` | | URL to poll for circuit breaking. Dynamic DynamoDB will only run if the circuit breaker returns `HTTP/200`. When polling the URL, the headers `x-table-name` and `x-gsi-name` will be sent identifying the table and GSI names, if applicable. |
| region | `str` | `us-east` | AWS region to use |

## Logging configuration

**Section name:** `[logging]`

| Option | Type | Default | Comment |
|---|---|---|---|
| log-file | `str` | | Path to log file. Logging to stdout if this option is not present |
| log-level | `str` | `info` | Log level (`debug`, `info`, `warning` or `error`) |
| log-config-file | `str` | | Path to external Python logging configuration file. Overrides both `log-level` and `log-file`. An example can be found in the Example configuration section. |

Dynamic DynamoDB will rotate the `log-file` nightly per default and keep 5 days of backups. If you want to override this behavior, please have a look at the `log-config-file` option which allows you to use custom Python logging configuration files.

## Table configuration

**Section name:** `[table: ^my_table$]`

Important note: The table name is treated as a regular expression. That means that `my_table` also will match `my_table2`, unless you express it as a valid regular expression; `^my_table$`. This feature enables you to easily configure many tables or tables with dynamic names.

| Option | Type | Default | Comment |
|---|---|---|---|
| allow-scaling-down-reads-on-0-percent | `bool` | `false` | Allow down-scaling of reads when 0% is used. |
| allow-scaling-down-writes-on-0-percent | `bool` | `false` | Allow down-scaling of writes when 0% is used. |
| always-decrease-rw-together | `bool` | `false` | Restrict scale down to only happen when both reads AND writes are in need of scaling down. Set this to `true` to minimize down-scaling. |
| circuit-breaker-timeout | `float` | 10000.00 | Timeout for the circuit breaker, in ms. Overrides the global setting if set. |
| circuit-breaker-url | `str` | | URL to poll for circuit breaking. Dynamic DynamoDB will only run if the circuit breaker returns `HTTP/200`. Overrides the global setting if set. When polling the URL, the header `x-table-name` will be sent identifying the table name. |
| decrease-reads-unit | `str` | `percent` | Set if we should scale down reads in `units` or `percent` |

Continued on next page

Table 4.1 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| decrease-reads-with | `int` | 50 | Number of `units` or `percent` we should scale down the read provisioning with. Choose entity with `decrease-reads-unit`. |
| decrease-writes-unit | `str` | `percent` | Set if we should scale down in `units` or `percent` |
| decrease-writes-with | `int` | 50 | Number of `units` or `percent` we should scale down the write provisioning with. Choose entity with `decrease-writes-unit`. |
| enable-reads-autoscaling | `bool` | `true` | Turn on or off autoscaling of read capacity. Deprecated! Please use `enable-reads-up-scaling` and `enable-reads-down-scaling` |
| enable-reads-down-scaling | `bool` | `true` | Turn on or off of down scaling of read capacity |
| enable-reads-up-scaling | `bool` | `true` | Turn on or off of up scaling of read capacity |
| enable-writes-autoscaling | `bool` | `true` | Turn on or off autoscaling of write capacity. Deprecated! Please use `enable-writes-up-scaling` and `enable-writes-down-scaling` |
| enable-writes-down-scaling | `bool` | `true` | Turn on or off of down scaling of write capacity |
| enable-writes-up-scaling | `bool` | `true` | Turn on or off of up scaling of write capacity |
| increase-consumed-reads-unit | `str` | `increase-reads-unit` | Set if we should scale up reads based on the consumed metric in `units` or `percent` |
| increase-consumed-reads-with | `int` | `increase-reads-with` | Number of `units` or `percent` we should scale up read provisioning based on the consumed metric |

Continued on next page

Table 4.1 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| increase-consumed-reads-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up read provisioning based on the consumption metric.<br>Detailed information on the scale dict can be found here.<br>If this is specified it will override `increase-consumed-reads-with` |
| increase-consumed-writes-unit | `str` | `increase-writes-unit` | Set if we should scale up writes based on the consumed metric in `units` or `percent` |
| increase-consumed-writes-with | `int` | `increase-writes-with` | Number of `units` or `percent` we should scale up write provisioning based on the consumed metric |
| increase-consumed-writes-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up write provisioning based on the consumption metric.<br>Detailed information on the scale dict can be found here.<br>If this is specified it will override `increase-consumed-writes-with` |
| increase-reads-unit | `str` | `percent` | Set if we should scale up reads in `units` or `percent` |
| increase-reads-with | `int` | 50 | Number of `units` or `percent` we should scale up the read provisioning with. Choose entity with `increase-reads-unit`. |
| increase-throttled-by-consumed-reads-unit | `str` | `increase-reads-unit` | Set if we should scale up reads based on throttled events with respect to consumption in `units` or `percent` |

Continued on next page

Table 4.1 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| increase-throttled-by-consumed-reads-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up read provisioning based on the throttled events with respect to consumption metric. Detailed information on the scale dict can be found here. |
| increase-throttled-by-consumed-writes-unit | `str` | `increase-writes-unit` | Set if we should scale up writes based on throttled events with respect to consumption in `units` or `percent` |
| increase-throttled-by-consumed-writes-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up write provisioning based on the throttled events with respect to consumption metric. Detailed information on the scale dict can be found here. |
| increase-throttled-by-provisioned-reads-unit | `str` | `increase-reads-unit` | Set if we should scale up reads based on throttled events with respect to provisioning in `units` or `percent` |
| increase-throttled-by-provisioned-reads-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up read provisioning based on the throttled events with respect to provisioning metric. Detailed information on the scale dict can be found here. |
| increase-throttled-by-provisioned-writes-unit | `str` | `increase-writes-unit` | Set if we should scale up writes based on throttled events with respect to provisioning in `units` or `percent` |

Continued on next page

Table 4.1 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| increase-throttled-by-provisioned-writes-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up write provisioning based on the throttled events with respect to provisioning metric. Detailed information on the scale dict can be found here. |
| increase-writes-unit | `str` | `percent` | Set if we should scale up in `units` or `percent` |
| increase-writes-with | `int` | 50 | Number of `units` or `percent` we should scale up the write provisioning with. Choose entity with `increase-writes-unit`. |
| lookback-window-start | `int` | 15 | Dynamic DynamoDB fetches data from CloudWatch in a window that streches between `now()-15` and `now()-10` minutes. If you want to look at slightly newer data, change this value. Please note that it might not be set to less than 1 minute (as CloudWatch data for DynamoDB is updated every minute). |
| lookback-period | `int` | 5 | Changes the duration of CloudWatch data to look at. For example, instead of looking at `now()-15` to `now()-10`, you can look at `now()-15` to `now()-14` |
| maintenance-windows | `str` | | Force Dynamic DynamoDB to operate within maintenance windows. E.g. `22:00-23:59, 00:00-06:00` |
| max-provisioned-reads | `int` | | Maximum number of provisioned reads for the table |

Continued on next page

Table 4.1 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| max-provisioned-writes | `int` | | Maximum number of provisioned writes for the table |
| min-provisioned-reads | `int` | | Minimum number of provisioned reads for the table |
| min-provisioned-writes | `int` | | Minimum number of provisioned writes for the table |
| num-read-checks-before-scale-down | `int` | 1 | Force Dynamic DynamoDB to have *x* consecutive positive results before scaling reads down (*1* means scale down immediately) |
| num-read-checks-reset-percent | `int` | 0 | Set a read consumption percentage when the *num-read-checks-before-scale-down* count should be reset. This option is optional, even if you use the *num-read-checks-before-scale-down* feature |
| num-write-checks-before-scale-down | `int` | 1 | Force Dynamic DynamoDB to have *x* consecutive positive results before scaling writes down (*1* means scale down immediately) |
| num-write-checks-reset-percent | `int` | 0 | Set a write consumption percentage when the *num-write-checks-before-scale-down* count should be reset. This option is optional, even if you use the *num-write-checks-before-scale-down* feature |
| reads-lower-alarm-threshold | `int` | | How many percent of the reads capacity should be used before trigging the low throughput alarm? |
| reads-lower-threshold | `int` | 30 | Scale down the reads with `--decrease-reads-with` if the currently consumed reads is as low as this percentage |

Continued on next page

Table 4.1 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| reads-upper-alarm-threshold | `int` | | How many percent of the reads capacity should be used before trigging the high throughput alarm? |
| reads-upper-threshold | `float` | 90 | Scale up the reads with `--increase-reads-with` if the currently consumed reads reaches this many percent |
| sns-message-types | `str` | | Comma separated list of message types to receive SNS notifications for. Supported types are `scale-up`, `scale-down`, `high-throughput-alarm` and `low-throughput-alarm` |
| sns-topic-arn | `str` | | Full Topic ARN to use for sending SNS notifications |
| throttled-reads-upper-threshold | `int` | 0 | Scale up the reads with `--increase-reads-with` if the count of throttled read events exceeds this count. Set to `0` (default) to turn off scaling based on throttled reads. |
| throttled-writes-upper-threshold | `int` | 0 | Scale up the writes with `--increase-writes-with` if the count of throttled write events exceeds this count. Set to `0` (default) to turn off scaling based on throttled reads. |
| writes-lower-alarm-threshold | `int` | | How many percent of the writes capacity should be used before trigging the low throughput alarm? |
| writes-lower-threshold | `int` | 30 | Scale down the writes with `--decrease-writes-with` if the currently consumed writes is as low as this many percent |
| writes-upper-alarm-threshold | `int` | | How many percent of the writes capacity should be used before trigging the high throughput alarm? |

Continued on next page

Table 4.1 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| writes-upper-threshold | `float` | 90 | Scale up the writes with `--increase-writes-with` if the currently consumed writes reaches this many percent |

## Global secondary index configuration

**Section name:** `[gsi: ^my_gsi$ table: ^my_table$]`

Important note: Both the GSI name and the table name is treated as regular expressions. That means that `my_gsi` also will match `my_gsi`, unless you express it as a valid regular expression; `^my_gsi$`. This feature enables you to easily configure many GSIs with one configuration section.

The `table:` section after `gsi:` **must** match with an existing `table:` section.

| Option | Type | Default | Comment |
|---|---|---|---|
| allow-scaling-down-reads-on-0-percent | `bool` | `false` | Allow down-scaling of reads when 0% is used. |
| allow-scaling-down-writes-on-0-percent | `bool` | `false` | Allow down-scaling of writes when 0% is used. |
| always-decrease-rw-together | `bool` | `false` | Restrict scale down to only happen when both reads AND writes are in need of scaling down. Set this to `true` to minimize down-scaling. |
| circuit-breaker-timeout | `float` | 10000.00 | Timeout for the circuit breaker, in ms. Overrides the global setting if set. |
| circuit-breaker-url | `str` | | URL to poll for circuit breaking. Dynamic DynamoDB will only run if the circuit breaker returns `HTTP/200`. Overrides the global setting if set. When polling the URL, the headers `x-table-name` and `x-gsi-name` will be sent identifying the table and GSI names. |
| decrease-reads-unit | `str` | `percent` | Set if we should scale down reads in `units` or `percent` |

Continued on next page

Table 4.2 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| decrease-reads-with | `int` | 50 | Number of `units` or `percent` we should scale down the read provisioning with. Choose entity with `decrease-reads-unit`. |
| decrease-writes-unit | `str` | `percent` | Set if we should scale down in `units` or `percent` |
| decrease-writes-with | `int` | 50 | Number of `units` or `percent` we should scale down the write provisioning with. Choose entity with `decrease-writes-unit`. |
| enable-reads-autoscaling | `bool` | `true` | Turn on or off autoscaling of read capacity. Deprecated! Please use `enable-reads-up-scaling` and `enable-reads-down-scaling` |
| enable-reads-down-scaling | `bool` | `true` | Turn on or off of down scaling of read capacity |
| enable-reads-up-scaling | `bool` | `true` | Turn on or off of up scaling of read capacity |
| enable-writes-autoscaling | `bool` | `true` | Turn on or off autoscaling of write capacity. Deprecated! Please use `enable-writes-up-scaling` and `enable-writes-down-scaling` |
| enable-writes-down-scaling | `bool` | `true` | Turn on or off of down scaling of write capacity |
| enable-writes-up-scaling | `bool` | `true` | Turn on or off of up scaling of write capacity |
| increase-consumed-reads-unit | `str` | `increase-reads-unit` | Set if we should scale up reads based on the consumed metric in `units` or `percent` |
| increase-consumed-reads-with | `int` | `increase-reads-with` | Number of `units` or `percent` we should scale up read provisioning based on the consumed metric |

Continued on next page

Table 4.2 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| increase-consumed-reads-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up read provisioning based on the consumption metric.<br>Detailed information on the scale dict can be found here.<br>If this is specified it will override `increase-consumed-reads-with` |
| increase-consumed-writes-unit | `str` | `increase-writes-unit` | Set if we should scale up writes based on the consumed metric in `units` or `percent` |
| increase-consumed-writes-with | `int` | `increase-writes-with` | Number of `units` or `percent` we should scale up write provisioning based on the consumed metric |
| increase-consumed-writes-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up write provisioning based on the consumption metric.<br>Detailed information on the scale dict can be found here.<br>If this is specified it will override `increase-consumed-writes-with` |
| increase-reads-unit | `str` | `percent` | Set if we should scale up reads in `units` or `percent` |
| increase-reads-with | `int` | 50 | Number of `units` or `percent` we should scale up the read provisioning with. Choose entity with `increase-reads-unit`. |
| increase-throttled-by-consumed-reads-unit | `str` | `increase-reads-unit` | Set if we should scale up reads based on throttled events with respect to consumption in `units` or `percent` |

Continued on next page

Table 4.2 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| increase-throttled-by-consumed-reads-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up read provisioning based on the throttled events with respect to consumption metric. Detailed information on the scale dict can be found here. |
| increase-throttled-by-consumed-writes-unit | `str` | `increase-writes-unit` | Set if we should scale up writes based on throttled events with respect to consumption in `units` or `percent` |
| increase-throttled-by-consumed-writes-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up write provisioning based on the throttled events with respect to consumption metric. Detailed information on the scale dict can be found here. |
| increase-throttled-by-provisioned-reads-unit | `str` | `increase-reads-unit` | Set if we should scale up reads based on throttled events with respect to provisioning in `units` or `percent` |
| increase-throttled-by-provisioned-reads-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up read provisioning based on the throttled events with respect to provisioning metric. Detailed information on the scale dict can be found here. |
| increase-throttled-by-provisioned-writes-unit | `str` | `increase-writes-unit` | Set if we should scale up writes based on throttled events with respect to provisioning in `units` or `percent` |

Continued on next page

Table 4.2 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| increase-throttled-by-provisioned-writes-scale | `dict` | | Dictionary containing threshold/increment key/value pairs. We should use this to scale up write provisioning based on the throttled events with respect to provisioning metric. Detailed information on the scale dict can be found here. |
| increase-writes-unit | `str` | `percent` | Set if we should scale up in `units` or `percent` |
| increase-writes-with | `int` | 50 | Number of `units` or `percent` we should scale up the write provisioning with. Choose entity with `increase-writes-unit`. |
| maintenance-windows | `str` | | Force Dynamic DynamoDB to operate within maintenance windows. E.g. `22:00-23:59, 00:00-06:00` |
| max-provisioned-reads | `int` | | Maximum number of provisioned reads for the table |
| max-provisioned-writes | `int` | | Maximum number of provisioned writes for the table |
| min-provisioned-reads | `int` | | Minimum number of provisioned reads for the table |
| min-provisioned-writes | `int` | | Minimum number of provisioned writes for the table |
| num-read-checks-before-scale-down | `int` | 1 | Force Dynamic DynamoDB to have *x* consecutive positive results before scaling reads down (*1* means scale down immediately) |

Continued on next page

Table 4.2 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| num-read-checks-reset-percent | `int` | 0 | Set a read consumption percentage when the *num-read-checks-before-scale-down* count should be reset. This option is optional, even if you use the *num-read-checks-before-scale-down* feature |
| num-write-checks-before-scale-down | `int` | 1 | Force Dynamic DynamoDB to have *x* consecutive positive results before scaling writes down (*1* means scale down immediately) |
| num-write-checks-reset-percent | `int` | 0 | Set a write consumption percentage when the *num-write-checks-before-scale-down* count should be reset. This option is optional, even if you use the *num-write-checks-before-scale-down* feature |
| reads-lower-alarm-threshold | `int` | | How many percent of the reads capacity should be used before trigging the low throughput alarm? |
| reads-lower-threshold | `int` | 30 | Scale down the reads with `--decrease-reads-with` if the currently consumed reads is as low as this percentage |
| reads-upper-alarm-threshold | `int` | | How many percent of the reads capacity should be used before trigging the high throughput alarm? |
| reads-upper-threshold | `float` | 90 | Scale up the reads with `--increase-reads-with` if the currently consumed reads reaches this many percent |

Continued on next page

Table 4.2 – continued from previous page

| Option | Type | Default | Comment |
|---|---|---|---|
| sns-message-types | `str` | | Comma separated list of message types to receive SNS notifications for. Supported types are `scale-up` , `scale-down`, `high-throughput-alarm` and `low-throughput-alarm` |
| sns-topic-arn | `str` | | Full Topic ARN to use for sending SNS notifications |
| throttled-reads-upper-threshold | `int` | 0 | Scale up the reads with `--increase-reads-with` if the count of throttled read events exceeds this count. Set to `0` (default) to turn off scaling based on throttled reads. |
| throttled-writes-upper-threshold | `int` | 0 | Scale up the writes with `--increase-writes-with` if the count of throttled write events exceeds this count. Set to `0` (default) to turn off scaling based on throttled reads. |
| writes-lower-alarm-threshold | `int` | | How many percent of the writes capacity should be used before trigging the low throughput alarm? |
| writes-lower-threshold | `int` | 30 | Scale down the writes with `--decrease-writes-with` if the currently consumed writes is as low as this many percent |
| writes-upper-alarm-threshold | `int` | | How many percent of the writes capacity should be used before trigging the high throughput alarm? |
| writes-upper-threshold | `float` | 90 | Scale up the writes with `--increase-writes-with` if the currently consumed writes reaches this many percent |

## Default configuration

**Section name:** `[default_options]`

Are you tired of setting the same configuration options for multiple tables or indexes? Then use the `[default_options]` section. It will let you create default values for all your tables and indexes. You can of course override those values by setting other values in your table or index specific configuration.

CHAPTER 5

# Example configuration files

## Example `dynamic-dynamodb.conf`

This is a full example of a Dynamic DynamoDB configuration file.

```
[global]
# AWS access keys
aws-access-key-id: AWS_ACCESS_KEY
aws-secret-access-key-id: AWS_SECRET_KEY

# AWS region to use
region: us-east-1

# How often should Dynamic DynamoDB monitor changes (in seconds)
check-interval: 300

# Circuit breaker configuration
# No provisioning updates will be made unless this URL returns
# a HTTP 2xx OK status code
#circuit-breaker-url: http://my.service.com/v1/is_up
#circuit-breaker-timeout: 500

[logging]
# Log level [debug|info|warning|error]
log-level: info

# Log file (comment out to get only console output)
log-file: /var/log/dynamic-dynamodb.log

# External Python logging configuration file
# Overrides both log-level and log-file
# log-config-file: /path/to/logging.conf

[table: ^my_table$]
#
```

```
# Read provisioning configuration
#

# Thresholds for trigging throughput alarm to send notification (%)
# reads-upper-alarm-threshold: 0
# reads-lower-alarm-threshold: 0

# Enable or disable reads autoscaling
enable-reads-autoscaling = true

# Thresholds for scaling up or down the provisioning (%)
reads-upper-threshold: 90
reads-lower-threshold: 30

# How many percent should Dynamic DynamoDB increase/decrease provisioning with (%)
increase-reads-with: 50
decrease-reads-with: 50

# Units to increase or decrease reads with, must be either percent or units
increase-reads-unit: percent
decrease-reads-unit: percent

# Maximum and minimum read provisioning
# Dynamic DynamoDB will not provision any more or less reads than this
min-provisioned-reads: 1
max-provisioned-reads: 500

#
# Write provisioning configuration
#

# Threshold for trigging throughput alarm to send notification (%)
# writes-upper-alarm-threshold: 0
# writes-lower-alarm-threshold: 0

# Enable or disable writes autoscaling
enable-writes-autoscaling = true

# Thresholds for scaling up or down the provisioning (%)
writes-upper-threshold: 90
writes-lower-threshold: 30

# How many percent should Dynamic DynamoDB increase/decrease provisioning with (%)
increase-writes-with: 50
decrease-writes-with: 50

# Units to increase or decrease writes with, must be either percent or units
increase-writes-unit: percent
decrease-writes-unit: percent

# Maximum and minimum write provisioning
# Dynamic DynamoDB will not provision any more or less writes than this
min-provisioned-writes: 1
max-provisioned-writes: 500

#
# Maintenance windows (in UTC)
#
```

```
#maintenance-windows: 22:00-23:59,00:00-06:00

#
# Simple Notification Service configuration
#

# Topic ARN to publish notifications to
#
# Example:
# sns-topic-arn: arn:aws:sns:us-east-1:123456789012:dynamic-dynamodb-my_table

# Message types to send as SNS notifications
#
# Comma separated list. Currently supported values:
# - scale-up                    Get notifications when the table is scaled up
# - scale-down                  Get notifications when the table is scaled down
# - high-throughput-alarm       Get notifications when exceed high throughput␣
↪threshold
# - low-throughput-alarm        Get notifications when below low throughput threshold
#
# Example:
# sns-message-types: scale-up, scale-down, high-throughput-alarm, low-throughput-alarm

#
# Other settings
#

# Allow down scaling when at 0% consumed reads
#allow-scaling-down-reads-on-0-percent: true
#allow-scaling-down-writes-on-0-percent: true

# Restrict scale down to only happen when BOTH reads AND writes are in need
# of scaling down. Set this to "true" to minimize down scaling.
#always-decrease-rw-together: true

[gsi: ^my_gsi$ table: ^my_table$]
#
# Read provisioning configuration
#

# Thresholds for trigging throughput alarm to send notification (%)
# reads-upper-alarm-threshold: 0
# reads-lower-alarm-threshold: 0

# Enable or disable reads autoscaling
enable-reads-autoscaling = true

# Thresholds for scaling up or down the provisioning (%)
reads-upper-threshold: 90
reads-lower-threshold: 30

# How many percent should Dynamic DynamoDB increase/decrease provisioning with (%)
increase-reads-with: 50
decrease-reads-with: 50

# Units to increase or decrease reads with, must be either percent or units
increase-reads-unit: percent
decrease-reads-unit: percent
```

```
# Maximum and minimum read provisioning
# Dynamic DynamoDB will not provision any more or less reads than this
min-provisioned-reads: 1
max-provisioned-reads: 500

#
# Write provisioning configuration
#

# Threshold for trigging throughput alarm to send notification (%)
# writes-upper-alarm-threshold: 0
# writes-lower-alarm-threshold: 0

# Enable or disable writes autoscaling
enable-writes-autoscaling = true

# Thresholds for scaling up or down the provisioning (%)
writes-upper-threshold: 90
writes-lower-threshold: 30

# How many percent should Dynamic DynamoDB increase/decrease provisioning with (%)
increase-writes-with: 50
decrease-writes-with: 50

# Units to increase or decrease writes with, must be either percent or units
increase-writes-unit: percent
decrease-writes-unit: percent

# Maximum and minimum write provisioning
# Dynamic DynamoDB will not provision any more or less writes than this
min-provisioned-writes: 1
max-provisioned-writes: 500

#
# Maintenance windows (in UTC)
#
#maintenance-windows: 22:00-23:59,00:00-06:00

#
# Simple Notification Service configuration
#

# Topic ARN to publish notifications to
#
# Example:
# sns-topic-arn: arn:aws:sns:us-east-1:123456789012:dynamic-dynamodb-my_table

# Message types to send as SNS notifications
#
# Comma separated list. Currently supported values:
# - scale-up                    Get notifications when the table is scaled up
# - scale-down                  Get notifications when the table is scaled
# - high-throughput-alarm       Get notifications when exceed high throughput␣
↪threshold
# - low-throughput-alarm        Get notifications when below low throughput threshold
#
# Example:
```

```
# sns-message-types: scale-up, scale-down, high-throughput-alarm, low-throughput-alarm

#
# Other settings
#

# Allow down scaling when at 0% consumed reads
#allow-scaling-down-reads-on-0-percent: true
#allow-scaling-down-writes-on-0-percent: true

# Restrict scale down to only happen when BOTH reads AND writes are in need
# of scaling down. Set this to "true" to minimize down scaling.
#always-decrease-rw-together: true
```

Note: The configuration of tables support regular expressions so you could write `[table: log_.* ]` if you want to target multiple tables with one config section, however if a table name matches the regex of more than one section, only the first match will be used. (This will let you set a `[table: .*]` section at the end as a catchall.)

## Example `logging.conf`

Below is an example of a logging configuration file used with the `--log-config-file` and `log-config-file` options. This kind of external logging configuration enables users to log through syslog, via custom log handlers or to other external services. It will also give control over logrotation and similar log management functions.

```
[loggers]
keys=root

[logger_root]
handlers=console,file
level=NOTSET

[formatters]
keys=default

[formatter_default]
format=%(asctime)s - %(name)s - %(levelname)s - %(message)s

[handlers]
keys=file,console

[handler_file]
class=handlers.TimedRotatingFileHandler
interval=midnight
backupCount=7
formatter=default
level=DEBUG
args=('/Users/sebastian/dynamic-dynamodb2.log',)

[handler_console]
class=StreamHandler
formatter=default
level=INFO
args=(sys.stdout,)
```

CHAPTER 6

# Command line options

Below is a listing of Dynamic DynamoDB's command line parameters.

```
usage: dynamic-dynamodb [-h] [-c CONFIG] [--dry-run] [--run-once]
                        [--check-interval CHECK_INTERVAL]
                        [--log-file LOG_FILE]
                        [--log-level {debug,info,warning,error}]
                        [--log-config-file LOG_CONFIG_FILE] [--version]
                        [--aws-access-key-id AWS_ACCESS_KEY_ID]
                        [--aws-secret-access-key AWS_SECRET_ACCESS_KEY]
                        [--daemon DAEMON] [--instance INSTANCE]
                        [--pid-file-dir PID_FILE_DIR] [-r REGION]
                        [-t TABLE_NAME]
                        [--reads-upper-threshold READS_UPPER_THRESHOLD]
                        [--throttled-reads-upper-threshold THROTTLED_READS_UPPER_
↪THRESHOLD]
                        [--reads-lower-threshold READS_LOWER_THRESHOLD]
                        [--increase-reads-with INCREASE_READS_WITH]
                        [--decrease-reads-with DECREASE_READS_WITH]
                        [--increase-reads-unit INCREASE_READS_UNIT]
                        [--decrease-reads-unit DECREASE_READS_UNIT]
                        [--min-provisioned-reads MIN_PROVISIONED_READS]
                        [--max-provisioned-reads MAX_PROVISIONED_READS]
                        [--num-read-checks-before-scale-down NUM_READ_CHECKS_BEFORE_
↪SCALE_DOWN]
                        [--num-read-checks-reset-percent NUM_READ_CHECKS_RESET_
↪PERCENT]
                        [--writes-upper-threshold WRITES_UPPER_THRESHOLD]
                        [--throttled-writes-upper-threshold THROTTLED_WRITES_UPPER_
↪THRESHOLD]
                        [--writes-lower-threshold WRITES_LOWER_THRESHOLD]
                        [--increase-writes-with INCREASE_WRITES_WITH]
                        [--decrease-writes-with DECREASE_WRITES_WITH]
                        [--increase-writes-unit INCREASE_WRITES_UNIT]
                        [--decrease-writes-unit DECREASE_WRITES_UNIT]
                        [--min-provisioned-writes MIN_PROVISIONED_WRITES]
                        [--max-provisioned-writes MAX_PROVISIONED_WRITES]
                        [--num-write-checks-before-scale-down NUM_WRITE_CHECKS_BEFORE_
↪SCALE_DOWN]
                        [--num-write-checks-reset-percent NUM_WRITE_CHECKS_RESET_
↪PERCENT]

Dynamic DynamoDB - Auto provisioning AWS DynamoDB

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        Read configuration from a configuration file
  --dry-run             Run without making any changes to your DynamoDB table
  --run-once            Run once and then exit Dynamic DynamoDB, instead of
                        looping
  --check-interval CHECK_INTERVAL
                        How many seconds should we wait between the checks
                        (default: 300)
  --log-file LOG_FILE   Send output to the given log file
  --log-level {debug,info,warning,error}
                        Log level to use (default: info)
  --log-config-file LOG_CONFIG_FILE
                        Use a custom Python logging configuration file.
                        Overrides both --log-level and --log-file.
  --version             Print current version number
  --aws-access-key-id AWS_ACCESS_KEY_ID
                        Override Boto configuration with the following AWS
                        access key
  --aws-secret-access-key AWS_SECRET_ACCESS_KEY
                        Override Boto configuration with the following AWS
                        secret key

Daemon options:
  --daemon DAEMON       Run Dynamic DynamoDB in daemon mode. Valid modes are
                        [start|stop|restart|foreground]
  --instance INSTANCE   Name of the Dynamic DynamoDB instance. Used to run
                        multiple instances of Dynamic DynamoDB. Give each
                        instance a unique name and control them separately
                        with the --daemon flag. (default: default)
  --pid-file-dir PID_FILE_DIR
                        Directory where pid file is located in. Defaults to
                        /tmp

DynamoDB options:
  -r REGION, --region REGION
                        AWS region to operate in (default: us-east-1
  -t TABLE_NAME, --table-name TABLE_NAME
                        How many percent should we decrease the read units
                        with?

Read units scaling properties:
  --reads-upper-threshold READS_UPPER_THRESHOLD
                        Scale up the reads with --increase-reads-with percent
                        if the currently consumed read units reaches this many
                        percent (default: 90)
  --throttled-reads-upper-threshold THROTTLED_READS_UPPER_THRESHOLD
                        Scale up the reads with --increase-reads-with percent
                        if the count of throttled read events exceeds this
                        count (default: 100)
  --reads-lower-threshold READS_LOWER_THRESHOLD
                        Scale down the reads with --decrease-reads-with
                        percent if the currently consumed read units is as low
                        as this percentage (default: 30)
  --increase-reads-with INCREASE_READS_WITH
                        How many percent should we increase the read units
                        with? (default: 50, max: 100)
  --decrease-reads-with DECREASE_READS_WITH
                        How many percent should we decrease the read units
                        with? (default: 50)
  --increase-reads-unit INCREASE_READS_UNIT
                        Do you want to scale in percent or units? (default:
                        percent)
  --decrease-reads-unit DECREASE_READS_UNIT
                        Do you want to scale in percent or units? (default:
                        percent)
  --min-provisioned-reads MIN_PROVISIONED_READS
                        Minimum number of provisioned reads
  --max-provisioned-reads MAX_PROVISIONED_READS
                        Maximum number of provisioned reads
  --num-read-checks-before-scale-down NUM_READ_CHECKS_BEFORE_SCALE_DOWN
                        Number of consecutive checks that must meet criteria
                        before a scale down event occurs
  --num-read-checks-reset-percent NUM_READ_CHECKS_RESET_PERCENT
                        Percentage Value that will cause the
                        num_read_checks_before scale_down var to reset back to
                        0

Write units scaling properties:
  --writes-upper-threshold WRITES_UPPER_THRESHOLD
                        Scale up the writes with --increase-writes-with
                        percent if the currently consumed write units reaches
                        this many percent (default: 90)
  --throttled-writes-upper-threshold THROTTLED_WRITES_UPPER_THRESHOLD
                        Scale up the reads with --increase-writes-with percent
                        if the count of throttled write events exceeds this
                        count (default: 100)
  --writes-lower-threshold WRITES_LOWER_THRESHOLD
                        Scale down the writes with --decrease-writes-with
                        percent if the currently consumed write units is as
                        low as this percentage (default: 30)
  --increase-writes-with INCREASE_WRITES_WITH
                        How many percent should we increase the write units
                        with? (default: 50, max: 100)
  --decrease-writes-with DECREASE_WRITES_WITH
                        How many percent should we decrease the write units
                        with? (default: 50)
  --increase-writes-unit INCREASE_WRITES_UNIT
                        Do you want to scale in percent or units? (default:
                        percent)
  --decrease-writes-unit DECREASE_WRITES_UNIT
                        Do you want to scale in percent or units? (default:
                        percent)
  --min-provisioned-writes MIN_PROVISIONED_WRITES
                        Minimum number of provisioned writes
  --max-provisioned-writes MAX_PROVISIONED_WRITES
                        Maximum number of provisioned writes
  --num-write-checks-before-scale-down NUM_WRITE_CHECKS_BEFORE_SCALE_DOWN
                        Number of consecutive checks that must meet criteria
                        before a scale down event occurs
  --num-write-checks-reset-percent NUM_WRITE_CHECKS_RESET_PERCENT
                        Percentage Value that will cause the
                        num_write_checks_before scale_down var to reset back
                        to 0
```

# CHAPTER 7

# Granular Scaling

The granular scaling feature in Dynamic DynamoDB allows users to specify fine tuning for up-scaling read and write provisioning by using the new optional `...-scale` config options.

This new config is specified by providing a dictionary of key-value pairs. The keys are the scaling thresholds for whichever metric is being evaluated (in percent) and the values are scaling amounts in either `units` or `percent` depending on the config specified for the associated `...-unit` config option.

If the `...-scale` config option is not specified then the scaling amount will come from the associated `...-with` config option (NOTE: This only applies to the `increase-consumed-reads-...` options. For the `increase-throttled-by-...` options, if the scale isn't specified then scaling based on these metrics will not occur).

If this option is specified then it will work as follows:

- If the metric being evaluated is at 0% then the scaling amount will be 0
- If the metric being evaluated is at a percentage in the range of scaling thresholds specified the scaling amount will be the value of the key-value pair whose key is equal to or lower than the current amount
- If the metric being evaluated is at a higher percentage than the highest threshold specified the scaling amount will be the the value of the final key-value pair

## Example

Config:

```
increase-consumed-reads-unit: percent
increase-consumed-reads-scale: {0: 0, 0.25: 5, 0.5: 10, 1: 20, 2: 50, 5: 100}
```

In this scenario:

- If the current consumed read units is up to 0.25% no scaling will occur.
- If the current consumed read units is at 0.25% or above but below 0.5% then read provisioning will be scaled up by 5%

- If the current consumed read units is at 0.5% or above but below 1% then read provisioning will be scaled up by 10%
- If the current consumed read units is at 1% or above but below 2% then read provisioning will be scaled up by 20%
- If the current consumed read units is at 2% or above but below 5% then read provisioning will be scaled up by 50%
- If the current consumed read units is at 5% or above read provisioning will be increased by 100%

CHAPTER 8

# IAM permissions

If you want to set up a separate IAM user for Dynamic DynamoDB, then you need to grant the user the following privileges:

- `cloudwatch:GetMetricStatistics`
- `dynamodb:DescribeTable`
- `dynamodb:ListTables`
- `dynamodb:UpdateTable`
- `sns:Publish` (used by the SNS notifications feature)

## Example IAM policy

Here's an example IAM policy. Please make sure you update the ARNs according to your needs.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:ListTables",
        "dynamodb:UpdateTable",
        "cloudwatch:GetMetricStatistics"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:*::dynamic-dynamodb"
      ]
    }
  ]
}
```

CHAPTER 9

# CloudFormation template

## Overview

To make it as easy as possible to get Dynamic DynamoDB up and running we provide a CloudFormation template. This template will launch an t1.micro instance with Dynamic DynamoDB pre-installed. All you need to do is to provide a Dynamic DynamoDB configuration file to use.

Please note that this will be charged towards your AWS account. The cost for a t1.micro server in `us-east-1` is less than 15 USD / month.

## Setup instructions

### Starting the CloudFormation stack

The following will create a new CloudFormation stack. This will launch a new EC2 instance.

1. Download the Dynamic DynamoDB CloudFormation template to your computer.
2. If you are already using Dynamic DynamoDB and have an existing configuration file, upload it to AWS S3. Take a note of the bucket name and path. You need to call the configuration `dynamic-dynamodb.conf`.
3. From the AWS CloudFormation dashboard click **Create stack**
4. On the **Template page**:
   (a) In the **Name** text box, enter the name of the stack. For example: `DynamicDynamoDB`.
   (b) In the **Template** section, click **Upload template file**. Click **Choose File** and select the file you downloaded in step 1.
   (c) Click **Next step**.
5. On the **Parameters page**:

   (a) In the **S3Bucket** text box, enter a URI for your Amazon S3 bucket. For example: `s3://bucket-name/dynamic-dynamodb/`. The URI **must** have a trailing slash (`/`). This should be the same bucket and path as you used in step 2. If you did not upload a template in step 2, choose any S3 bucket and path.
   (b) Define the region for the S3 bucket in the **S3BucketRegion** text box. This is needed due to a limitation in the AWS CLI (https://github.com/aws/aws-cli/issues/564).
   (c) In the **KeyPair** text box, enter the name of your Amazon EC2 key pair
   (d) (Optional) Change the instance type. Default is t1.micro.
   (e) Click **Next Step**.
6. On the **Options page**, click **Next Step**. There are no tasks to perform on this page.
7. On the **Review page**, review the options for your stack
8. If you are OK with all configuration, click **Create**

CloudFormation will now create your stack, it will take a few minutes. You can follow the progress by watching the **Events** tab on your stack.

## Debugging template errors

If you need to debug errors in the template setup, you'll want to disable CloudFormation rollbacks. You can do that when you deploy the CloudFormation template in the section above. On the **Options page** (step 6), click **Advanced** and select **No** under **Rollback on failure**.

This will keep the EC2 instance running and allow you to log in and debug any issues you might experience.

## Accessing the EC2 instance

You can find your instance in the EC2 instance list. The instance name is `dynamic-dynamodb`.

You can then access the instance via SSH:

```
ssh -i /path/to/key.pem ec2-user@<hostname>
```

## Updating configuration

You can update the configuration directly on the EC2 instance. Simply modify `/etc/dynamic-dynamodb/dynamic-dynamodb.conf` and restart Dynamic DynamoDB.

## Starting and stopping Dynamic DynamoDB

Dynamic DynamoDB runs in daemon mode on the EC2 instance.

You can start the daemon by running:

```
service dynamic-dynamodb start
```

And you can stop it by running:

```
service dynamic-dynamodb stop
```

## Deleting the stack

If you wish to remove the CloudFormation stack, follow these steps:

1. From the AWS CloudFormation dashboard, select your stack.
2. Click **Delete stack** and then **Yes, delete**, when prompted.

# CHAPTER 10

# Release notes

## 2.5.1

**Release date** 2017-08-12

- Fix *logutils* version to 0.3.3 to prevent pip bug (#324).

## 2.5.0

**Release date** 2017-08-10

- Prevents scaling down write capacity as long as there are throttled write requests (#319).
- Print GSI changes as integers instead of floats (#323).
- Log table name together with the GSI name for uniqueness (#331).
- Documentation fixes (#329)

## 2.4.0

**Release date** 2016-11-25

- Add *–show-config* option (#317). Thanks for the pull request michaelkaye

## 2.3.1

**Release date** 2016-08-03

- Bug fix: Include *HTTP 200* as an acceptable response code for the circuit breaker

## 2.3.0

**Release date** 2016-07-16

- Accept any HTTP *2xx* code as response for a healthy circuit breaker (#292)
- Allow *]* in configuration section names (#281)
- Security fix for pid and log files (#301)
- Update CloudFormation template to allow dots in key-pair name (#287)
- Update CloudFormation template to add support for China (Beijing) region (#288)
- Fixed *allow-scaling-down-on-0-percent* bug (#304)

## 2.2.1

**Release date** 2015-12-18

- Make sure we aren't downscaling if disabled in the config (#279). Thanks for the pull request Sazpaimon

## 2.2.0

**Release date:** 2015-10-23

- Circuit breaker a per-table/gsi config option (#276). Thanks for the pull request @Sazpaimon
- Fix minor spelling and formatting issues (#278)

## 2.1.2

**Release date:** 2015-10-05

- Fix the lower_threshold checking (#275). Thanks for the pull request @KyleAlanDavis

## 2.1.1

**Release date:** 2015-10-02

- Depend on argparse for Python 2.6 (#274)
- Fix import of ordered dicts for Python 2.6 (#273). Thanks for the pull request @KyleAlanDavis
- Fix upscaling reads and writes for GSI (#271). Thanks for the pull request @qqshfox

## 2.1.0

**Release date:** 2015-09-30

- Support for granular downscaling (#263). Thanks for the pull request @Sazpaimon

- Allow configurable lookback period instead of the hardcoded 5 minute one (#261). Thanks for the pull request @Sazpaimon
- Don't scale down if it would scale below the current consumed units (#268). Thanks for the pull request @Sazpaimon

## 2.0.2

**Release date:** 2015-09-30

- Use HTTP rather than SSH for git clone (#266). Thanks for the pull request @superprat

## 2.0.1

**Release date:** 2015-09-29

- Fix value error when logging floats (#265)
- Fix installation on Python < 2.7 (#260). Thanks for the pull request @Sazpaimon

## 2.0.0

**Release date:** 2015-08-26

- Match each table to at most one config section (#250). Thanks for the pull request @memory
- Granular up-scaling and new scaling metrics (*#252 <https://github.com/sebdah/dynamic-dynamodb/issues/250>*). Thanks for the pull request @omnidavez

Please note that this change is not backwards compatible due to the nature of *#250 <https://github.com/sebdah/dynamic-dynamodb/issues/250>*__. Though it is in a minority of use cases, it may in some cases break existing configuration. Details can be found in the pull request #251

## 1.20.6

**Release date:** 2015-06-02

- Logging an AccessDeniedException in dynamodb.update_table_provisioning causing a KeyError (#247)

## 1.20.5

**Release date:** 2015-04-25

- Update cloudformation template to work around pip permissions issue (#238). Thanks for the pull request @jasonrdsouza

## 1.20.4

**Release date:** 2015-03-17

- Does not scale down when one of the provisoned values is already at the lowest limit (#237). Thanks for the pull request @lcabral37

## 1.20.3

**Release date:** 2015-02-17

- Bug in percent calculation for CUs (#235)

## 1.20.2

**Release date:** 2014-12-23

- Output failed to show floats and crashed the daemon (#229). Thanks @nickveenhof for the pull request.

## 1.20.1

**Release date:** 2014-12-22

- Updated retrying version to 1.3.3 (#218)

## 1.20.0

**Release date:** 2014-12-22

- Faster Up-scaling Based on Consumed Read/Write (#227)

Thanks @pragnesh for the help with this release

## 1.19.2

**Release date:** 2014-11-29

- Remove 100% increase limit in configuration validation (#225)

## 1.19.1

**Release date:** 2014-11-19

- Fixed bug when disabling read and writes up/down scaling separately (#221)

## 1.19.0

**Release date:** 2014-10-16

- Support for DynamoDBs new flexible scaling (#207)
- Fixed a bug with consecutive checks for GSIs (#206)
- Typo in log message when auto scaling of table writes was disabled (#209)

## 1.18.5

**Release date:** 2014-09-26

- Weird behavior when a single table is matched by multiple regexes (#203)

## 1.18.4

**Release date:** 2014-09-25

- Tables should automatically scale up if below their minimums (#202)

## 1.18.3

**Release date:** 2014-08-04

- Throttling up scaling was ignored if a regular down scale occurred (#198)
- Read config setting used in wrong situation for GSIs (#199)

## 1.18.2

**Release date:** 2014-07-23

- Throughput alarms does not honor lookback windows (#197)

## 1.18.1

**Release date:** 2014-07-18

- Incorrect division in consumption calculation (#195)

## 1.18.0

**Release date:** 2014-07-17

- Make the CloudWatch lookback window start point configurable (#192)
- Make it possible to turn on and off up-scaling or down-scaling (#147)

- Enhance boto logging for GSI errors (#194)
- Automatically bump the doc version (#191)

## 1.17.0

**Release date:** 2014-06-23

- No credentials needed in the CloudFormation template, an IAM Instance Profile will be automatically created
- The CloudFormation template will now launch Amazon Linux 2014.03.1 AMIs
- Minor fixes to make the CloudFormation template work smoother
- Fallback to use boto authentication (#188)
- Handle logging configuration exceptions (#189)

## 1.16.0

**Release date:** 2014-06-11

- SNS notifications when throughput thresholds are exeeded. (#174). Thanks (@Jofen) for the pull request!

## 1.15.1

**Release date:** 2014-06-05

- Locked `boto` version to 2.28.0 (#183)

## 1.15.0

**Release date:** 2014-06-02

- Default options for tables and GSIs is now supported using the *[default_options]* section (#181)

## 1.14.0

**Release date:** 2014-05-21

A special thanks to (@ulsa) for his help with this release.

- Document SNS permission requirements (#171)
- Wrong region in log message for instance profile authentication (#170)
- Improved logging for daemon commands (#165). Pull request from @ulsa
- Removed inconsistent debug printout (#164). Pull request from @ulsa
- Fixed some minor doc issues (#172). Pull request from @ulsa
- Added column for default values (#173). Pull request from @ulsa

- Fixed issues with docs for command-line options (#176). Pull request from @ulsa
- Implement max retry count for CloudWatch metrics fetching (#178)
- throttled-reads-upper-threshold is documented as being a percentage; actually a count (#169)
- Catch permission denied when creating pid file (#167)
- No error message when having insufficient IAM permissions (#166)
- Docs for -t incorrect (#161)
- Create example IAM policy (#177)

## 1.13.1

**Release date:** 2014-05-16

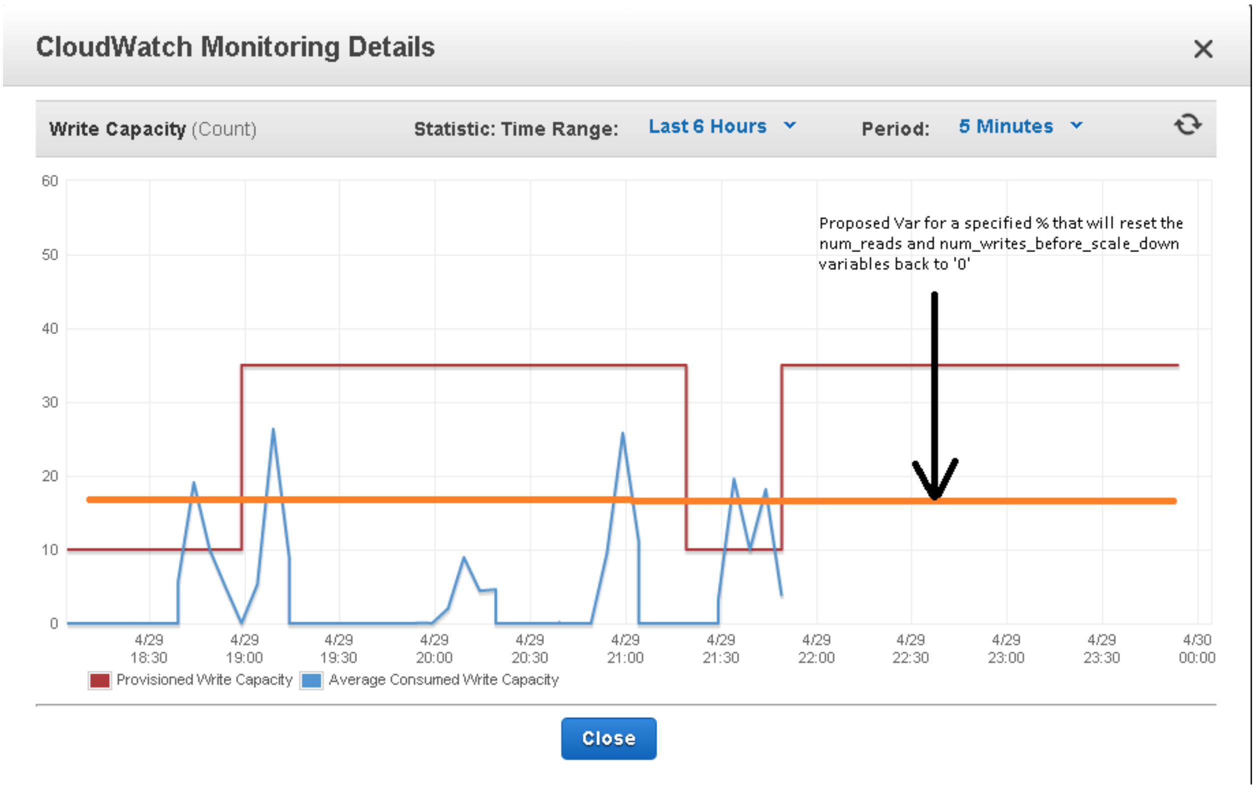
- Bugfix: Fix a few provisioning update issues (#160). Thanks (@Jofen) for the pull request!

## 1.13.0

**Release date:** 2014-05-08

A special thanks to (@johnbartels) for his help with the consecutive checks feature.

- It is now possible to scale down after *x* consecutive checks where Dynamic DynamoDB have seen a need for scaling down. This will make it even easier to target the times during the day where you want to scale down. See the `num-read-checks-before-scale-down` and `num-write-checks-before-scale-down` options. (#148)
- In addition to the consecutive checks feature we have support for resetting the consecutive checks counter on certain consumption percentages using `num-read-checks-reset-percent` and `num-write-checks-reset-percent`. In the example below we would be able to avoid the second scale down by setting a reset percentage that was slightly higher than the consumption spikes.



- You can now run Dynamic DynamoDB once instead of looping using the `--run-once` command line option (#152)
- Merged pull request: Fixed regex wildcard example (#151) by (@tayl0r)
- Merged pull request: Fix log message when autoscaling of writes has been disabled (#150) by (@alexkuang)

## 1.12.1

**Release date:** 2014-04-28

- Fixed reading of wrong config options

## 1.12.0

**Release date:** 2014-04-26

- Scaling up will now be allowed even outside maintenance windows (#138)
- Reduced code duplication between GSI and table scaling. Implemented some tests for the core calculatations (#139)

## 1.11.0

**Release date:** 2014-04-14

- You can now turn on and off scaling for reads and writes separately (#137)

- Make it possible to set pid file location (#146)

## 1.10.7

**Release date:** 2014-04-01

- Implemented back off strategy when retrieving CloudWatch metrics (#134)

## 1.10.6

**Release date:** 2014-04-01

- Fixed config parsing (#143)

## 1.10.4 + 1.10.5

**Release date:** 2014-03-31

- Fixed issue with throtting that overrides regular scaling rules (#142)

## 1.10.3

**Release date:** 2014-03-31

- Fixed bug in default option parsing for GSIs (#141)

## 1.10.2

**Release date:** 2014-03-29

- Throttling should only be checked for if explicitly configured (#135)
- Catching exception in metrics fetching (#134)
- *always-decrease-rw-together* blocked scaling up (#133)

## 1.10.1

**Release date:** 2014-03-24

- Fixed typo in SNS notification subjects
- Merged GSI fixes (#131)
- Updated readme with IAM information (#132)

## 1.10.0

**Release date:** 2014-03-21

- Added support for SNS notifications when Dynamic DynamoDB updates table provisioning (#104)
- Dynamic DynamoDB can now authenticate using EC2 instance profiles (#128)
- Enhanced error handling of unknown exceptions
- Bug fix: `check-interval` was not properly set when restarting the daemon (#130)

## 1.9.1

**Release date:** 2014-03-17

- Daemon mode does not reload the table / GSI lists #127
- dynamic-dynamodb crashes when a DynamoDB table is deleted during execution #126
- Catch invalid regular expressions in the configuration #125

## 1.9.0

**Release date:** 2014-03-07

- Introducing support for throttled request mitigation #120. Many thanks to @msh9 for his pull request!

## 1.8.3

**Release date:** 2014-02-27

- Fixes in configuration and CloudFormation template

## 1.8.2

**Release date:** 2014-02-21

- The dyn-run tag is not visible in log files #120

## 1.8.1

**Release date:** 2014-02-20

- Python dictConfig is not available in Python 2.6 #119

## 1.8.0

**Release date:** 2014-02-19

- Add support for external logging configuration file #74
- logging through syslog #49

## 1.7.3

**Release date:** 2014-01-28

- circuit breaker option bailing out w/ exception #105

## 1.7.2

**Release date:** 2014-01-23

- CU increases fail if decreases fail due to exceeded limits #103

## 1.7.1

**Release date:** 2014-01-04

- Rounding increase values up #100
- Fixed bug with configuration for multiple tables #101. Thanks @msh9!

## 1.7.0

**Release date:** 2013-12-26

- Added support for global secondary indexes #73
- Fixed Log level configuration in configuration file is overridden #75
- Recursively retrieve all tables #84. Submitted by @alxmrtn
- Loop dynamic-dynamodb in command line mode #91
- *Migrated to ‘‘boto.dynamodb2‘* #72 <https://github.com/sebdah/dynamic-dynamodb/issues/72>‘__
- Describe configuration options in the documentation #80
- Move release notes to the documentation #79
- Better exception handling fixed in #96, #97 and #98
- Silently skipping configured tables that does not exist in DynamoDB #94
- Enhanced configuration validation #93

A full list of closed issues can be found here

Special thanks to the AWS DynamoDB for their support with this release.

## 1.6.0

**Release date:** 2013-11-21

- Documented project in Sphinx - http://dynamic-dynamodb.readthedocs.org
- Fixed Failure on non-matching regular expressions #69
- Fixed bug cleanup logs in case of noop updates #71 - Thanks [@tmorgan](https://github.com/tmorgan)

## 1.5.8

**Release date:** 2013-10-18

- Fixed bug Under some circumstances Dynamic DynamoDB crashes when table is deleted #68

## 1.5.7

**Release date:** 2013-10-17

- Closed Support for running dynamic-dynamodb with supervisord #66 with Pull Request #67. Thanks @pragnesh!

## 1.5.6

**Release date:** 2013-10-06

- Fixed issue where DDB didn't support increasing capacity two times #65

## 1.5.5

**Release date:** 2013-08-31

- Change log level of informational message

## 1.5.4

**Release date:** 2013-08-29

- *Added missing ‘‘key_name‘* parameter <https://github.com/sebdah/dynamic-dynamodb/issues/60>‘__

## 1.5.3

**Release date:** 2013-08-27

- Added missing sleep statement fixes

## 1.5.2

**Release date:** 2013-08-27

- *Issue with ‘‘always-decrease-rw-together‘* option <https://github.com/sebdah/dynamic-dynamodb/issues/55>‘__
- ListTables permission required

The AWS `ListTables` permission is no longer a hard requirement. It’s only needed if you’re using regular expressions to configure your DynamoDB tables.

## 1.5.1

**Release date:** 2013-08-22

- No module named core (fixed by #54)

Fixed bug in the 1.5.0 release.

## 1.5.0

**Release date:** 2013-08-22

- Support for regular expressions in config

Thanks [@pragnesh](https://github.com/pragnesh) for adding this feature!

## 1.4.0

**Release date:** 2013-08-14

- Retrying failed DynamoDB connections

## 1.3.6

**Release date:** 2013-07-21

- int() argument must be a string or a number, not ‘NoneType’ (#50)

## 1.3.5

**Release date:** 2013-06-17

- increase_writes_unit parameter is used while it should be decrease_writes_unit (#47)

## 1.3.4

**Release date:** 2013-06-13

- An attempt to update provisioning is made even if the requested values are equal to the tables current values (#46)

## 1.3.3

**Release date:** 2013-06-08

- Increasing to a minimum provisioned throughput value doesn't take into account the current table's throughput (#45)
- dynamic-dynamodb –version causes AttributeError in cli (#44)

## 1.3.2

**Release date:** 2013-05-14

- increase_reads_in_percent calculations are incorrect (#40)

## 1.3.1

**Release date:** 2013-05-10

- Fix Python 2.6 support (#39)

## 1.3.0

**Release date:** 2013-05-01

This Dynamic DynamoDB release makes it possible to use multiple Dynamic DynamoDB instances in parallel in daemon mode. Simply use the `--instance` flag to separate the difference instances with a unique name. Then control them as usual with the `--daemon` flag.

- Allow to run multiple instances in parallel (#37)

## 1.2.5

**Release date:** 2013-04-29

- Handle ResourceInUseException better (#36)
- Add –log-level option to command line (#34)

## 1.2.4

**Release date:** 2013-04-26

- Mix up between percent and units (#35)
- Broken build fixed

## 1.2.0

**Release date:** 2013-04-20

- Add support for dead-man's-switch API call (#25)

## 1.1.0

**Release date:** 2013-04-17

- Update provisioning in units not just percentage (#22)
- Increase in percent does not add to current provisioning properly (#33)
- Enhance configuration option validation (#32)

## 1.0.1

**Release date:** 2013-04-17

- Minor fix: Ugly output removed

## 1.0.0

**Release date:** 2013-04-16

The 1.0.0 release is a leap forward when it comes to module structure and extendability. Please remember that this is still Release in beta in PyPI, so all bug reports are very welcome. File any odd behavior or bugs in GitHub Issues.

- Restructure the Dynamic DynamoDB code base (#30)
- Support multiple tables in one configuration file (#19)
- Change pid file name (#31)
- Handle combinations of configuration file and command line options better (#24)

## 0.5.0

**Release date:** 2013-04-12

- Add –log-file command line option (#20)
- Allow scale down at 0% consumed count (#17)

- "only downscale reads AND writes" option would be useful (#23)

## 0.4.2

**Release date:** 2013-04-11

- Unhandled exception: ValidationException (#28)
- Handle DynamoDB provisioning exceptions cleaner (#29)

## 0.4.1

**Release date:** 2013-04-10

- No logging in –daemon mode (#21)

## 0.4.0

**Release date:** 2013-04-06

- Support for daemonizing Dynamic DynamoDB (#11)
- Enhanced logging options (#4)
- Add –version flag to dynamic-dynamodb command (#18)

## 0.3.5

**Release date:** 2013-04-05

- Handle missing table exceptions (#12)
- Bug fix: No upscaling happening when scaling limit is exceeded (#16)

## 0.3.4

**Release date:** 2013-04-05

- Bug fix: Min/max limits seems to be read improperly from configuration files (#15)

## 0.3.3

**Release date:** 2013-04-05

- Bug fix: Mixup of read and writes provisioing in scaling (#14)

## 0.3.2

**Release date:** 2013-04-05

- Bug fix: Improper scaling under certain circumstances (#13)

## 0.3.1

**Release date:** 2013-04-04

- Bug fix: ValueError: Unknown format code 'd' for object of type 'str' (#10)

## 0.3.0

**Release date:** 2013-03-27

This release contains support for configuration files, custom AWS access keys and configurable maintenance windows. The maintenance feature will restrict Dynamic DynamoDB to change your provisioning only during specific time slots.

- Add support for configuration files (#6)
- Configure AWS credentials on command line (#5)
- Support for maintenance windows (#1)

## 0.2.0

**Release date:** 2013-03-24 - First public release

## 0.1.1

**Release date:** 2013-03-24 - Initial release

# CHAPTER 11

# Bugs and feature requests

Please report any found bugs or file feature requests at our GitHub issues page.

# CHAPTER 12

# License

CHAPTER 13

# Dynamic DynamoDB

AWS DynamoDB is a great service, but it lacks support for automated throughput scaling. This is where Dynamic DynamoDB enters the stage. It provides automatic read and write provisioning for DynamoDB.

## Features in short

- Scale up and down DynamoDB automatically
- Restrict scaling to certain time slots
- Monitor multiple DynamoDB tables at the same time
- Gives you control over how much reads and writes you want to scale up and down with
- Dynamic DynamoDB has support for max and min limits so that you always knows how much money you spend at most and how much capacity you can be guaranteed
- Get notifications when your table provisioning changes via e-mail, HTTP, SQS etc (via AWS SNS)
- Support for circuit breaker API call. If your service is experiencing disturbances, Dynamic DynamoDB will not scale down your DynamoDB tables

## Basic usage

This example will configure Dynamic DynamoDB to:

- Scale up your DynamoDB table when the consumed reads 90% of the total provisioned reads
- Scale up your DynamoDB table when the consumed writes 90% of the total provisioned writes
- Scale up your reads with 50%
- Scale up your writes with 40%
- Scale down your DynamoDB table when the consumed reads 30% of the total provisioned reads

- Scale down your DynamoDB table when the consumed writes 40% of the total provisioned writes
- Scale down your reads with 40%
- Scale down your writes with 70%
- Check for changes every 5 minutes

Command:

```
dynamic-dynamodb --table-name my-table \
                 --reads-upper-threshold 90 \
                 --reads-lower-threshold 30 \
                 --increase-reads-with 50 \
                 --decrease-reads-with 40 \
                 --writes-upper-threshold 90 \
                 --writes-lower-threshold 40 \
                 --increase-writes-with 40 \
                 --decrease-writes-with 70 \
                 --check-interval 300
```

Please note that using configuration files instead of command line options will give you even more control over the service.

## Author

This project is maintained by Sebastian Dahlgren (GitHub | Twitter | LinkedIn)