# Tag Group Application

(Note: please refer to [this link](https://github.com/rjgleave/aws-tag-groups) to see this full document, with images)

This solution will allow you to define tag groups, which can be used to automatically apply multiple tags to resources which are identified with a specific tag key. The lambda uses the AWS Resource Group tagging API to efficiently search for resources and apply groups of tags to them. This service provides powerful filtering capability and will automatically be updated by AWS as new services are added to the platform. It also provides a powerful API for applying groups of tags to groups of resources with a single operation.

This design includes support for pagination, to support scaling to very large numbers of resources. There are also two options for performing the update: 1) to decouple the tag update operation from the query, using SQS, or 2) perform an immediate update along with the query. Option 1 is preferred, in order to limit API throttling. The architecture is shown below.

![Tag Group Architecture](data:text/html; charset=utf-8;base64,)

Tag Group Architecture

## What’s Here

This repo includes:

1. README.md - this file
2. FOLDER: dynamo - this contains code to help build the tag group table in dynamoDB. It includes:
   * createTagGroupTable.py - a program to create the base dynamo table
   * DataEntryTemplate.xlsx - an Excel template with sample data for the dynamo table
   * sample-data.csv - a CSV extract of the excel file above
   * sample-data.json - a JSON equivalent to the CSV file above
   * loadTagGroupTable.py - a program to load the dynamodb table with sample data
3. FOLDER: policy - contains a json file for defining a custom policy: custom-tag-group-policy.json
4. FOLDER: tag-builder-direct - It includes:
   * tagbuilder.py - the main lambda program to update resources with tags.
5. FOLDER: tag-builder-lambda - It includes:
   * tagbuilder-paginate.py - same as lambda program above, except can be called from command line.
6. FOLDER: api-throttler - contains the lambda which is triggered by SQS. This performs the call to the AWS ResourceGroupsAPI to update tags if you are operating in de-coupled mode (see the SQS-ENABLED environment variable below)

## Setup Instructions

1. Enter sample data into the spreadsheet: DataEntryTemplate.xlsx (note: column B below auto-computes) ![Data Entry Template](data:text/html; charset=utf-8;base64,)
2. Export your data from the spreadsheet into a CSV file named: sample-data.csv
3. Convert the CSV file to JSON format. The output file should be named: sample-data.json. You can do this with any online site such as: http://www.convertcsv.com/csv-to-json.htm
4. Run the program to create the dynamodb table: createTagGroupTable.py
5. Run the program to load the data into the new table: loadTagGroupTable.py
6. Verify that the tagGroup table has been created in dynamoDB
7. Create new custom policies, using the examples in the policy FOLDER: custom-tag-group-policy.json and custom-sqs-resource-api-policy.json
8. Create a service role for your main lambda (tagbuilder.py). It should include the following policies:
   * CloudwatchActionsEc2Access
   * AWSResourceGroupsReadOnlyAccess
   * AmazonDynamoDBReadOnlyAccess
   * AmazonDynamoDBFullAccesswithDataPipeline
   * AWSLambdaBasicExecutionRole
   * ResourceGroupsandTagEditorFullAccess
   * custom-tag-group-policy (step #7 above)

![Custom Tagging Role](data:text/html; charset=utf-8;base64,)

Custom Tagging Role

1. Create another service role for your SQS Lambda. It should include the following policies:
   * custom-sqs-resource-policy (step #7 above)
   * all of the policies indicated in the other role above.

![Custom SQS Role](data:text/html; charset=utf-8;base64,)

Custom SQS Role

1. Install the lambda program (tagbuilder.py in the tag-builder-lambda folder) in your chosen region. Configuration details below:
   * Use the lambda role defined above
   * Create a trigger using Cloudwatch Events
   * Define the Cloudwatch Event with your lambda as a target. The event should include a schedule of whatever frequency you prefer.
   * Create Environment Variables in the AWS console for the lambda. (see image below)
     + GROUP\_KEY: used set identify the main Tag Key which controls tagging. It is used to filter resources. The value can be whatever you want to use for the the group name, however it must match the key in the dynamoDB table.
     + TAG\_FILTER: used to turn on and off tag filtering. The value must be True or False. If the filter is ON (true) then it will only update resources which have missing or incorrect tags. If the filter is OFF (false), then this process will update ALL resources with new tags.
     + SQS\_ENABLED: used to change the update mode from immediate to decoupled/queued. The value must be True or False. If set to True, then all tagging requests will be queued into SQS messages and the AWS ResourceGroupTaggingAPI will be invoked by a separate Lambda which reads the SQS records. If set to False, the AWS ResourceGroupTaggingAPI will be invoked immediately (based on volume of updates, this may result in throttled API requests/errors)
     + AWS\_REGION: used to set the region. The value must be a valid region name.

![Tag Group Update Lambda](data:text/html; charset=utf-8;base64,) ![Environment Variable](data:text/html; charset=utf-8;base64,)

1. Create an SQS Queue to accept tag request messages from tagbuilder.py. This should be set to FIFO.

![SQS Message Queue](data:text/html; charset=utf-8;base64,)

SQS Message Queue

1. Install the Lambda program to read the SQS queue (qprocessor.py in the api-throttler folder).
   * Create a trigger from SQS. Select the SQS Queue which was created above.
   * Use the SQS service role created above.

![SQS Queue Processor Lambda](data:text/html; charset=utf-8;base64,)

SQS Queue Processor Lambda

**Additional Resources**

Blog: The New Resource Groups Tagging API Makes It Easier to Programmatically Manage Tags on Resources Across AWS Services: https://aws.amazon.com/about-aws/whats-new/2017/03/the-new-resource-groups-tagging-api-makes-it-easier-to-programmatically-manage-tags-on-resources-across-aws-services/