CloudWatch for MarkLogic

CloudWatch for MarkLogic helps set up AWS CloudWatch monitoring and alarms for MarkLogic Server

Quick Start

The ideal setup would use the standard MarkLogic AMI, which is built on AWS Linux. Setting up your MarkLogic cluster on AWS either manually or using AWS CloudFormation is described here

On any host in your cluster

- *sudo yum -y install git*
- *git clone git@github.com:mustard57/cloud-watch-for-marklogic.git*
- *cd cloud-watch-for-marklogic*
- Modify your settings in *config.py* (see below)
- *./cronCloudWatchUpdate.sh* (adds metric storing as a cron job)
- *python update-cloudwatch-metrics.py* --setAlarm (sets up alarms)

config.py

This file holds the config required to retrieve monitoring data from MarkLogic, label appropriately in AWS and configure email based alerting. It ships as below

```
USER="cloudwatch-monitoring-user"
PASSWORD="cloudwatch-monitoring-user"
HOST="localhost"
SERVER_NAME="cloudwatch-demo-rest"
SERVER_DATABASE="cloudwatch-demo-content"
EMAIL_FOR_SNS="youremail@yourdomain.com"
```

Configure as follows

USER - user id used to retrieve monitoring data from MarkLogic. Must have the *manage-user* role
PASSWORD - password for the above user
HOST - MarkLogic host that will be used to access server metrics. If running these scripts on a host in the cluster, *localhost* is fine
SERVER_NAME - Name of the *application* server you would like to monitor
SERVER_DATABASE - Name of the *database* you would like to monitor
EMAIL_FOR_SNS - When setting alarms up, *update-cloudwatch-metrics.py* creates an SNS topic with name *SERVER_NAME*and subscribes *this address* to it. So configure with the address you would like alerts sent to. **Note that** this address **will get a confirmation email from AWS which must be responded to if you want to receive alerts**

Shipped config can be used without modification with the demonstration application below. You will need to create your own monitoring user if not.

cloudwatch-demo

It's nice to know it works, so CloudWatch for MarkLogic ships with an application, **cloudwatch-demo** that allows you to easily simulate a running application and gather some meaningful metrics.

About cloudwatch-demo

cloudwatch-demo supplies a MarkLogic rest extension endpoint on port 8006 (by default). *http://YOUR_HOST:8006/LATEST/resources/content* called as a GET request will retrieve a randomly chosen number of documents (1 - 10) while the same resource called as a PUT will create 1 - 10 documents. The documents are 500 - 1000 words long (randomly chosen), with the words being chosen from the file found in */data/available-vocabulary.xml*. The database is initially seeded with 1000 such documents. Just to keep things exciting, the MarkLogic caches get cleared on average every 40 requests. All these constants configurable via */lib/constants.xqy* . It is designed to be used by an application user - cloudwatch-demo-user as a matter of good practice.

Installation

- *cd test-application*
- Edit *deploy/local.properties* setting user and password properties to a user/password combination that allows application setup
- *./doAll.sh* (runs ml local bootstrap / deploy modules / deploy content - see Roxy for details)
- If running a cluster, run *./ml local create_application_replica_forests* which will replicate application forests (and you will get replication related metrics)

Simulation via JMeter

You can simulate activity via JMeter. Install JMeter and then open jmeter/simulation-for-cloudwatch.jmx. Click 'Cloudwatch data simulation', left hand side. You will need to change the host parameter to a hostname on which you have installed cloudwatch-demo. Best is to provide the name of the Elastic Load Balancer associated with your cluster (automatically provided when using MarkLogic CloudFormation). You may wish to change the thread count.

Removal

Run *python update-cloudwatch-metrics.py --deleteAlarm* to remove alarms. This will also remove the SNS topic and subscription created when alarms were set.

If you would like to uninstall the test application, run *ml local wipe* followed by *ml local restart* should you wish to re-use the port.

The Small Print

Dependencies

Requires python 2.7 which is found on AWS Linux by default. Makes use of the boto library which again is installed by default on AWS Linux.

Costs

https://aws.amazon.com/cloudwatch/pricing/ is definitive. It currently says

$0.5 per metric per month
$0.1 per alarm per month

This must be pro-rated in some way in that I've created hundreds of custom metrics while writing this, but my monthly costs are pretty small (c. $25 for the month and I think that's mostly EC2). Custom metric data rolls off after a fortnight. Alarms will hang around unless you delete them (which you can do manually, or use the script above).

Unavailable metrics

If you don't have replica forests or aren't using database replication some metrics won't be available. Cloudwatch For MarkLogic won't try and store metrics that don't exist and won't set alarms that don't exist. The flip side is that if you do bring these things online, re-run *python update-cloudwatch-metrics.py --setAlarm* so alarms get configured. Also don't run --setAlarm **before** you install your app otherwise most things will be unavailable.

Your mileage may vary

Metrics and thresholds for alarms may be found in the file *metrics.xml* which is based on the configuration used by the MarkLogic Nagios Plugin You may wish to choose your own values, or make use of metrics that the standard configuration does not use. Also note that for some alarms alerts are raised when *deviating from current values* e.g. host or database count. The corollary of that is that you need to re-run --setAlarm if expected values change.

Debugging

Run *python update-cloudwatch-metrics.py* with the --debug flag set and it will tell you what it's doing without actually doing it. In particular, run with the --setMetrics flag to see what values are being stored. *crontab -l* will show you that this is the flag used by the cron job.