CLOUD WATCH

* Cloud watch monitors your aws resources and applications in aws for real time. Cloud watch collect and track metrics of your aws resources an send notifications regarding changes in resources based on your rule that you defined.
* For example, you can monitor cpu usage, disk usage of ec2 instances and based on this usage, we can scale resources up and down when we need.
* It is basically a metrics repository. Aws resources like ec2 instances put metrics into the repository and retrieve statistics based on that repository.
* You can create cloud watch alarms to notify you when certain criteria are met.
* Cloud watch refresh data intervals every 5 minutes which is free of cost. If you enable **detailed montoring**, it will give you 1 minute refresh intervals which is paid.
* To create a billing alarm, go to Billing, enable billing alerts in preferences,

**Click billing (left).**

**Click create alarm.**

**Select an emails (separated with ,).**

**Enter an amount ($10).**

**Click Create alarm.**

* A conformation mail sent to your mail. Confirm that mail to activate billing alarm.
* The alarm we created will monitor your account billing. If the billing goes past $10, aws will send you a mail.

CREATING CUSTOM LOGS

# By default, there will be a lot of cloud watch metrics for all the resources in aws. Useful metrics like CPU Utilization, disk read & write, network in and out etc. But, some metrics are not available by default like disk space, memory utilization.

# So, for this we have to create custom metrics in our EC2 instances and need to make an API call from EC2 instances to send the data to cloud watch to monitor those resources.

# First, Install Perl Modules.

# yum install perl-Switch perl-DateTime perl-Sys-Syslog perl-LWP-Protocol-https perl-Digest-SHA --enablerepo="rhui-REGION-rhel-server-optional" -y

# Install unzip package.

# yum install zip unzip

# Download cloudwatchmonitoringscripts.zip file. And extract it. Aws-scripts-mon dir will be extracted from that file.

# Go to aws-scripts-mon dir. There are some files,

# Cloudwatchclient.pm = shared Perl module calls cloudwatch from other scripts.

# Mon-put-instance-data.pl = collects system info like mem, disk, swap and sends it to cloudwatch.

# Mon-get-instance-data.pl = queries cloudwatch to send instance statistics on which the script is executed.

# Awscreds.template = contains aws secret and access keys.

# License.txt = contains apache 2.0 license

# Notice.txt = copyright notice.

# In that dir, create a file called awscreds.conf and copy the awscreds.template to awscreds.conf. mention your aws access key and secret access key in awscreds.conf.

# If you want to use IAM role instead of aws credentials. Make sure that you have gave these permissions for that role..

# cloudwatch:PutMetricData

# cloudwatch:GetMetricStatistics

# cloudwatch:ListMetrics

# ec2:DescribeTags

# After configuring credentials, be in that directory (aws-scripts-mon) to configure custom logs.

# Now type this command to check the scripts. It will test the scripts without sending any data to cloud watch.

# ./mon-put-instance-data.pl --mem-util --verify --verbose

# If no errors occurred, then it is successfully written.

# Now, type this cmd to send ram and disk metrics to cloudwatch.

# ./mon-put-instance-data.pl --mem-util --mem-used --mem-avail --disk-space-util --disk-path=/

# In my system, the partition is mounted in / path. If your partition is mounted elsewhere than “/”, then specify the exact mount point of the partition in the diskpath=mountedpoint.

# Wait for few minutes and refresh the cloudwatch console, you will see disk, memory metrics for your instance.

# Used = displays used space in GB… --mem-used, --swap-used …

# Util= displays data in percentage (%)...--mem-util, --swap-util, --disk-space-util…

# Avail = displays available data in GB…--mem-avail, --disk-space-avail.

# To collect metrics for an ASG and send it to cloudwatch without reporting individual instance metrics,

# ./mon-put-instance-data.pl --mem-used --mem-avail --auto-scaling=only

# You can do this cmd in cron job. Go to crontab –e file,

# \*/1 \* \* \* \* cmd-of-cloudwatch --from-cron

# Now after setting cron job for custom metrics, it will send the data to cloud watch for every time period as you specified in the job.

# After adding the custom metrics you can’t delete them from cloud watch console, those will be automatically deleted from console after 2 weeks, once the instance is terminated.

CLOUD WATCH ALARMS

* You can create alarms to watch over single metric. The alarm performs actions based on your settings over a specific time periods. The actions can be EC2 actions, auto scaling action, notification send to a sns topic.
* For ex, if your EBS is idle for 2 days, you can set an alarm with cloudwatch that, if your EBS is idle for specific time over specific time periods, send a notification to these recipients.
* To create an alarm. Go to Cloud watch, click alarm, choose create alarm.

**Choose a Metric topic (EC2, EBS, S3, SNS).**

**Select EC2 & Choose metric (cpu, disk i/o etc).**

**Click, Next.**

**Type a name for the alarm.**

**Description for the alarm.**

**type % of cpu.**

**Type, after how many time periods you should receive mail.**

**Select alarm state = ok, alarm, insufficient.**

**Ok** = Alarm state is within the thresold.

**Alarm** = State is breached the specified threshold.

**Insufficient** = Alarm still didn’t got the data of the resource.

**Select time period.**

**Type topic name.**

**Type mails to get notified.**

* To take EC2 action like stop, terminate, reboot for that instance, click EC2 Action in alarm page.
* Select an action...Stop, terminate, reboot. Create an IAM role for that work and click Create Alarm.

# CLOUDWATCH LOGS

* It is used to monitor application logs and system logs of ec2 instances in cloud watch.
* We have to install an cloud watch logs agent in EC2 instances and mention which logs we need to send to cloudwatch.
* When you send logs to cloudwatch, it sends them to log group which is nothing but a container for set of log files. Log group has retention settings as how long you want to save the data in it.
* First, you have to install awslogs package. If you are using amazon linux instance, awslogs package will be present by default.
* Install awslogs package = **yum install awslogs**
* Go to **/etc/awslogs/awscli.conf**,

**Add your region** and **secret access keys**.

* Start the service = **service awslogs start.**
* If you are using redhat, centos, ubuntu images. You have to install cloud watch agent manually.
* You have to download the logs agent from the internet and install the agent.
* To install logs agent, follow these steps.
* Download the cloud watch logs agent = **curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py -O**
* Download the agent dependencies = **curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/AgentDependencies.tar.gz -O**
* Untar the dependency package = **tar -xvf AgentDependencies.tar.gz -C /tmp/**
* Specify your region in the place of **“us-east-1”,** while installing the logs agent.
* Install the cloud watch logs agent = **sudo python ./awslogs-agent-setup.py --region us-east-1 --dependency-path /tmp/AgentDependencies**
* After installing logs agent it will ask you to mention aws access keys and secret access keys and region.
* **Path of log file to upload** = mention you path to monitor.
* **Destination Log Group name** = give a name to log group.
* **Choose Log Stream name** = use instance id (number=1).
* **Choose Log Event timestamp format** = select time format.
* **Choose initial position of upload** = start (or) end of the file.
* **More log files to configure** = yes (or) No.
* After configuring all these options, wait for a few minutes and refresh the logs page to monitor your application logs in cloud watch console.
* You can also use IAM role instead of using access key credentials. Create an role and give full cw access to that role And attach that role while creating ec2 instances.
* You can’t add IAM role for running instance. At that time, you can use access keys.
* Now, you will see log group in aws cloud watch console. You can see each and every log in that log group.
* You can also create log group separately for various applications and services.
* Go to **/var/awslogs/etc/awslogs.conf**, at the end of line copy last 7 lines and change them accordingto your requirement.

**[/var/log/messages]**

**datetime\_format = %b %d %H:%M:%S**

**file = /var/log/messages**

**buffer\_duration = 5000**

**log\_stream\_name = {instance\_id}**

**initial\_position = start\_of\_file**

**log\_group\_name = /var/log/messages**

CLOUDTRAIL

* It is used to monitor API calls made by all users in an aws account.
* It stores all logs in S3 buckets that we mentioned while creating cloudtrail. It saves logs with sourceip, timestamp, region, action occurred, username and instance id. So, we can find easily who did what actions with this cloudtrail.
* You can see all the activities in event activity of last 7 days in aws console. As in that console, they mention ip, username, action, instance id etc.
* Go to cloudtrail console,

**Type a trail name**

**Select Bucket(to store logs)**

**Apply to all regions = yes (or) No.**

**Click Save.**

* Now, go to S3 and check, there will be folders created in the bucket that we mentioned while creating trail. All the logs are store here.
* You can integrate cloud trail with cloud watch logs to monitor api calls from aws console.
* You can integrate cloud trail with cloud watch logs only after creating a trail.
* Go to cloud trail, Select an existing trail.
* In cloud watch logs Section,

**Click, Configure.**

**Create new log group (or) select existing group.**

* A cloud watch logs role and cloud watch logs policy will be created automatically on behalf of ourselves. Select the role and policy in next page.

**Click, View details. And Select.**

**IAM Role = cloud watch logs role.**

**Policy Name = Cloud watch logs.**

**Click Allow.**

* Wait for few minutes, you will see a log group storing all your API calls from cloud trail.