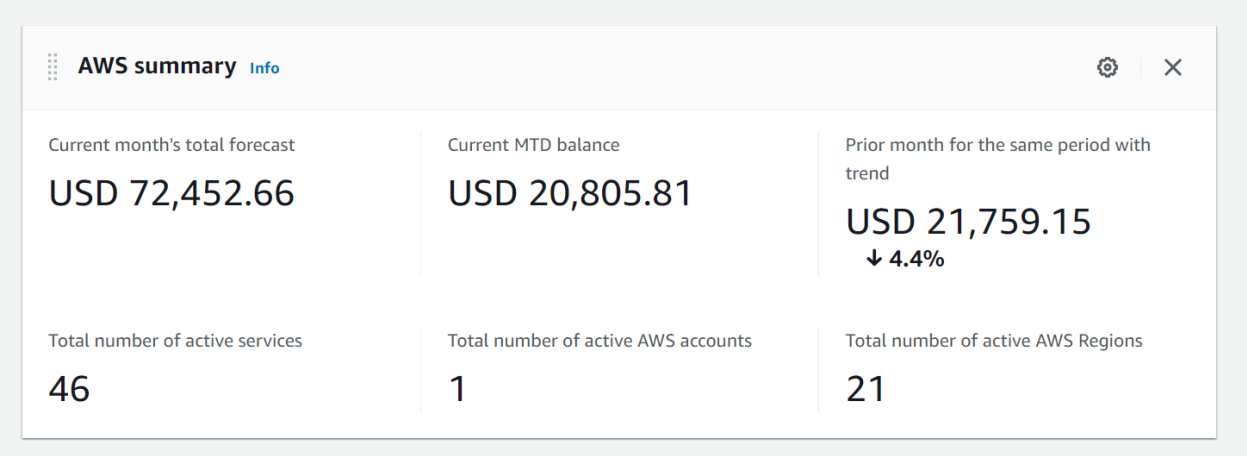
CLOUDWATCH LOG ANALYSIS

US east OHIO has highest cost.

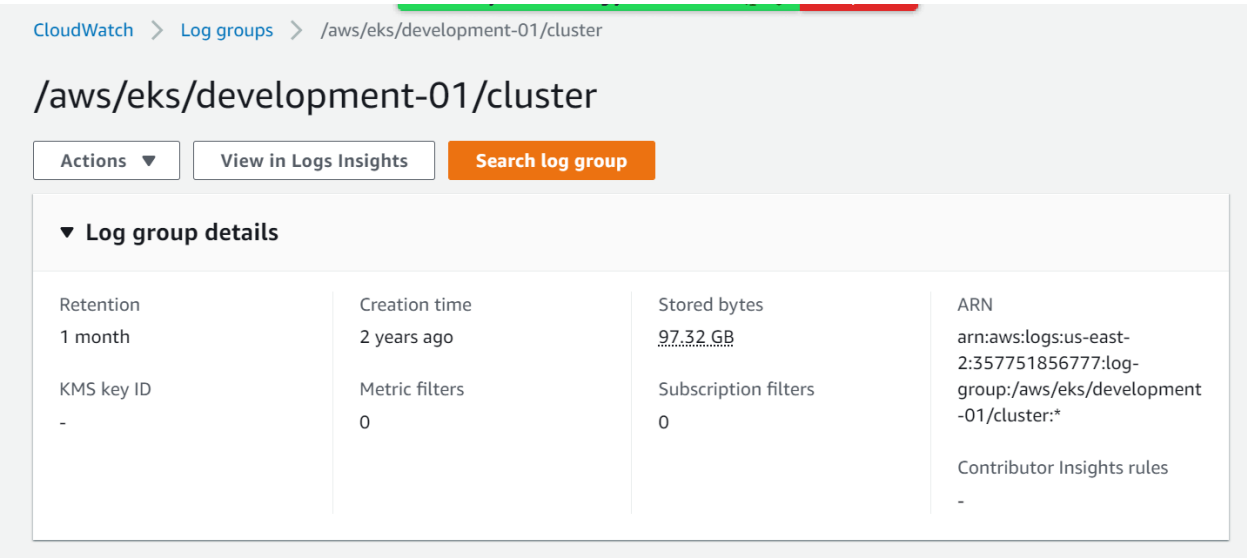


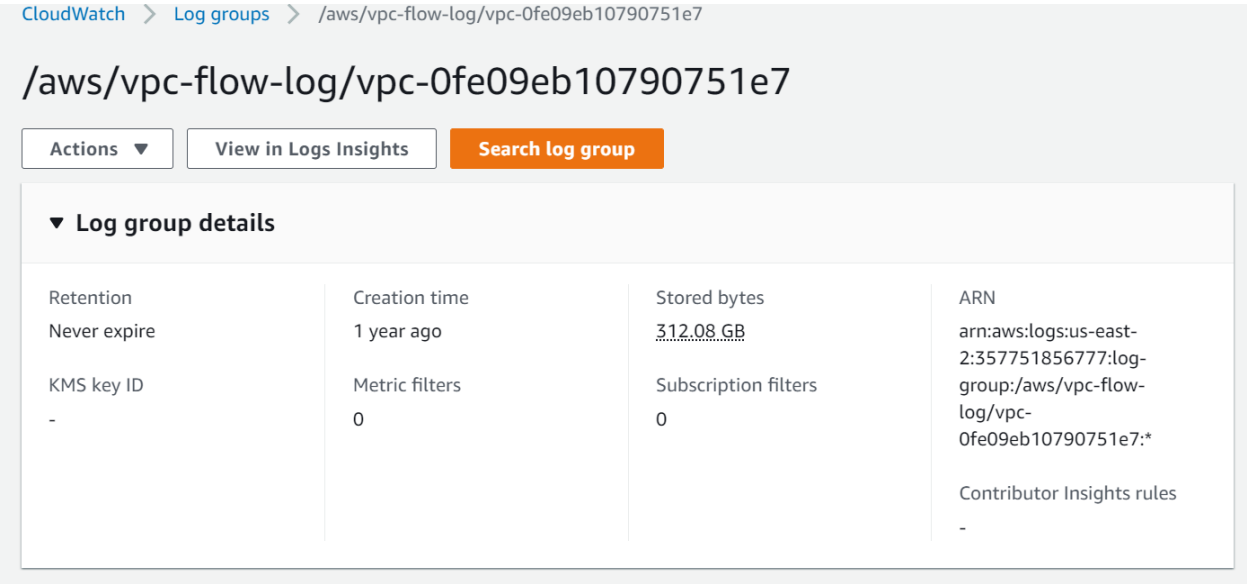
▾ CloudWatch	$6,970.19

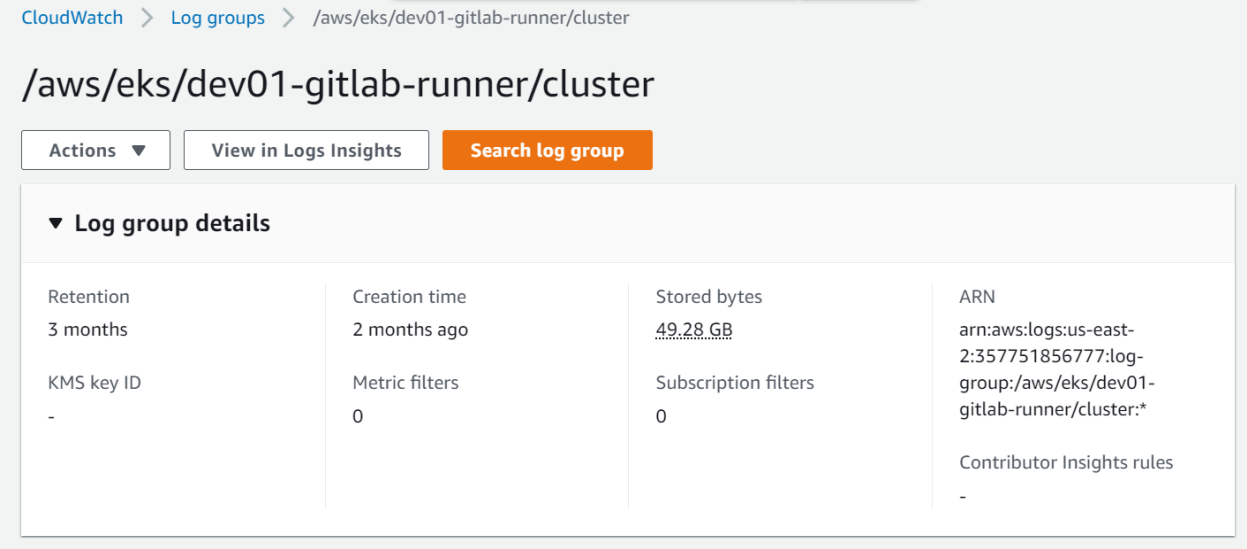
Amazon CloudWatch		**$2,403.05**
$0.0012 per canary run per month - US East (Ohio)	1,192,493.000 Runs	$1,430.99
$0.01 per 1,000 metrics requested using GetMetricData API - US East (Ohio)	194,968.000 Metrics	$1.95
$0.01 per 1,000 requests	3,388,798.000 Requests	$33.89
$0.03 per million Events for CW:ContributorEventsManaged in US East (Ohio)	27.000 Events	$0.00
$0.10 per alarm metric month (standard resolution) - US East (Ohio)	1,090.983 Alarms	$109.10
$0.30 per alarm metric month (high resolution) - US East (Ohio)	10.954 Alarms	$3.29
$0.30 per metric-month for the first 10,000 metrics - US East (Ohio)	2,745.021 Metrics	$823.51
$0.50 per rule per month for CW-Rule-DynamoDB in US East (Ohio)	0.664 Rule-Hour	$0.33
AmazonCloudWatch PutLogEvents		**$3,689.44**
$0.50 per GB log data ingested - US East (Ohio)	3,795.340 GB	$1,897.67
$0.50 per GB log data ingested for the first 10TB - US East (Ohio)	3,583.544 GB	$1,791.77
AmazonCloudWatch StartQuery		**$294.08**
$0.005 per GB log data scanned by CloudWatch Logs Insights queries - US East (Ohio)	58,816.828 GB	$294.08
AmazonCloudWatch USE2-TimedStorage-ByteHrs		**$34.14**
$0.03 per GB-mo of log storage - US East (Ohio)	1,138.044 GB-Mo	$34.14

Log groups using high archival storage (Almost 95%)

Period: 11 may to 11 june







Tips to reduce costs

Detailed monitoring:

1) Charges are incurred by detailed CloudWatch monitoring for EC2 instances, Auto Scaling group launch configurations, or API gateways.

2) To reduce costs, turn off detailed monitoring of instances, Auto Scaling group launch configurations, or API gateways.

Custom metrics:

1) Charges are incurred by monitoring more than 10 custom metrics. Custom metrics are metrics created manually or collected by CloudWatch agent installed on EC2 instances.

2) To reduce costs, turn off monitoring of custom metrics. To show custom metrics only, enter **NOT AWS** in **Search for any metric, dimension or resource ID** box of the CloudWatch console.

Cloudwatch alarms:

1)_Charges are incurred by the number of metrics associated with a CloudWatch alarm. For example, if you have a single alarm with multiple metrics, you're charged for each metric.

2) To reduce costs, remove unnecessary alarms.

Cloudwatch logs:

1) Charges are incurred by ingestion, archival storage, and analysis of Amazon CloudWatch Logs.

2) To reduce ingestion costs, you can re-evaluate logging levels and eliminate the ingestion of unnecessary logs i.e. to remove unnecessary application logging like console.log from nodejs services.

3) Archival charges are related to the log storage costs over time. The retention policy determines how long CloudWatch Logs keeps the data. To reduce storage costs, consider changing the retention policy (for example, you can set the retention policy to keep data for 1 week, 1 month, and so on).

- There is one issue with retention policy. After the retention time, logs are deleted.
- AWS allows you to export logs to S3 before retention period is over. With this approach we could save cost by moving the logs to S3 IA / S3 IA One zone / Glacier by setting proper lifecycle policy.

4) Analysis charges occur when Log Insights is used to query logs. To reduce analysis charges, queries that can be run in shorter time should be used.

Cloudwatch dashboards:

1) Charges are incurred when we use more than 3 dashboards (with up to 50 metrics). Currently I guess we have more than 50 dashboards.

2) To reduce costs, we need to delete unnecessary dashboards. We need to keep the total number of metrics across all dashboards to less than 50.

Cloudwatch synthetics:

1) We_can use Amazon CloudWatch Synthetics to create canaries. Canaries use selenium webdriver in a headless chrome to check endpoint availability and latency. It also takes screenshot UI. This adds to the cost. Canaries use lambda functions written in node/python. Charges are incurred when we use more than 100 canary runs/month.

2) To reduce costs, delete canaries for unnecessary endpoints.

Cloudwatch GetMetricData API calls:

1) **GetMetricData** api call does not come under free tier so its usage will incur cost.

2) To reduce costs make GetMetricData calls through the console for free rather than making them through the AWS CLI/ SDK. And retrieve only those metrics that are actively monitored. So less data retrieval means less cost.