УДК 004.6-024

REAL-TIME AWS RESOURCES MONITORING AND ANALYTICS

C.S.Dzik
Graduate student BSUIR, Utech Solutions, software and data engineer

I.I. Piletski,
PhD, Associate Professor of Informatics Department of the BSUIR

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus.
E-mail: kanstantind@gmail.com, ianmenski@gmail.com.

C. S. Dzik

Graduate student BSUIR, Utech Solutions, software and data engineer, conduct scientific research of anomaly detection using autoencoder artificial neural network.

I. I. Piletski PhD,

Associate Professor of the department of Informatics Department of Belarusian State University of Informatics and Radioelectronics. In the field of IT for over 49 years. Participation in the development of several dozen large projects: chief designer of the project, chief architect of software and information support, project manager, head of department, head of laboratory (Scientific Research Institute of Computer Science, Academy of Sciences of Belarus, IBA, BSUIR).

Abstract. In this article provides an approach for real-time AWS resources monitoring and analytics. The main is centralizing logs, events, and metrics for cloud-native applications running on Amazon Web Services (AWS). This work presents approach how the following AWS services can be utilized together: Amazon CloudWatch, CloudTrail, Kinesis, Elasticsearch, Lambda, Simple Cloud Storage, EC2.

Keywords: monitoring, analytics, cloud, AWS, CloudWatch, Kinesis, Elasticsearch, Lambda, S3, EC2.

Introduction.

Currently, the main problem with visibility in cloud services such as AWS is that data comes from different sources on each layer of the application stack, varying in format, frequency and importance, all of which must be tracked in real time by the appropriate roles in the organization. Thus, when scaling a particular product and organization, an AWS solution for centralized logging is critical. A focused effort to centralize all disparate monitoring data, regardless of the source of the data, is needed to ensure full visibility of this kind. This paper focuses on centralizing logs, events and metrics for cloud-native applications running on Amazon Web Services (AWS). AWS certainly has a full suite of cloud services, with 175 services as of 2021. Each AWS service uses its own set of metrics, events and logs. In addition, there are performance metrics created by applications running in AWS. To centralize this data, AWS provides CloudWatch. Because CloudWatch is an AWS built-in service, there is little or no configuration required, and CloudWatch automatically records some default monitoring data from many AWS services immediately upon activation. AWS Kinesis Stream provides a platform for consuming said data from various sources and processing it for analysis purposes. AWS Kinesis Stream transfers all monitoring data from CloudWatch to AWS Elasticsearch, as it has a more powerful data analytics engine.

The goal of this work is to propose some technology for developing applications based on the sharing of AWS services that operate in real time with large amounts of data.

Materials and Methods.

There are many types of logs in AWS; the more applications and services used in AWS, the more complex the logging needs will be. The lack of visibility into logs, events, and metrics creates operational challenges in the use of the cloud by modern applications.

If the logs are so valuable, why can't we just grep through them to find what we need? It turns out it is not that simple, and there are several reasons for that.

1) First, the logs capacity, to be searched, is enormous and growing.

2) Second, when looking at a single log, there is only one data slice, such as an application data slice or a server data slice. Without the ability to correlate application data slices across multiple logs from different components, it is difficult to get a complete picture of the problems that arise.

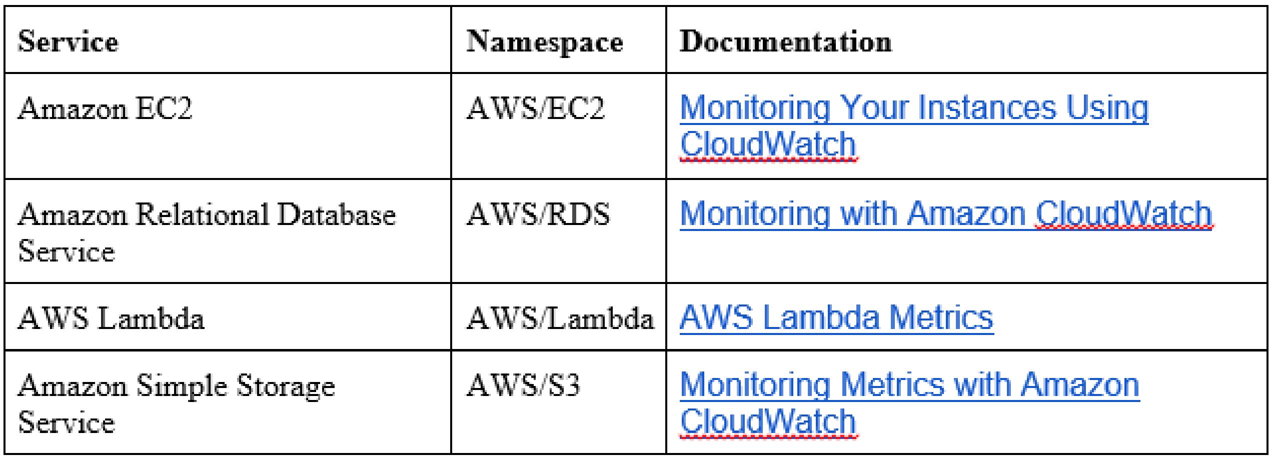
3) Third, companies are required by certain regulations to keep all logs and keep developers away from production machines. In this scenario, a centralized real-time log management solution satisfies both of these requirements.

The two primary sources of logs are applications running on AWS services and AWS services themselves. CloudWatch is the primary log collector, collecting logs and metrics on application performance and service usage. Separately, AWS stores all API calls made to the AWS service within CloudTrail.

Once the data is collected in CloudWatch and CloudTrail, it is ready for analysis and alert configuration. The data can then be archived in AWS S3 or sent to a separate logging and monitoring solution.

Here is a list of the most vital AWS services and what you should consider when setting up logging with CloudWatch. The following examples of AWS services publish metrics to CloudWatch [8]. For information about the metrics and dimensions, see the specified documentation (see fig. 1).

Why we need to ingest and process logs in real time data? The field of Big Data is undergoing radical changes. Processing huge amounts of data in real time has become the new norm, and to stay competitive, companies simply have to process incoming data in minutes or seconds. Systems often create hundreds or thousands of events per second, so it is important to know what is going on in your application. The key difference is the ability to import and interpret events in real time. Our approach provides information within seconds of its occurrence, alerting the developer of warning signs that indicate a problem is occurring and increasing the chances of identifying, diagnosing and resolving problems before they negatively impact end users (fig. 1.) [9].



Service	Namespace	Documentation
Amazon EC2	AWS/EC2	Monitoring Your Instances Using CloudWatch
Amazon Relational Database Service	AWS/RDS	Monitoring with Amazon CloudWatch
AWS Lambda	AWS/Lambda	AWS Lambda Metrics
Amazon Simple Storage Service	AWS/S3	Monitoring Metrics with Amazon CloudWatch

Figure 1. Examples of popular AWS Services that publish CloudWatch metrics

Approach and Results.

Amazon Kinesis is a service provided by AWS for real-time data processing. In the scenario we are about to discuss, the events coming from CloudWatch to Kinesis are a stream, where Kinesis will perform all kinds of functions based on our requirements. There are many other solutions in the community for handling data streams. Apache Kafka, Apache Spark are just some examples of solutions currently in use in the community. The main reason we chose Kinesis is that our Elasticsearch solution will be built as an AWS service, and since all the servers are fully managed by AWS, it is really easy for us to focus more on results than infrastructure management [2] (see fig. 2).

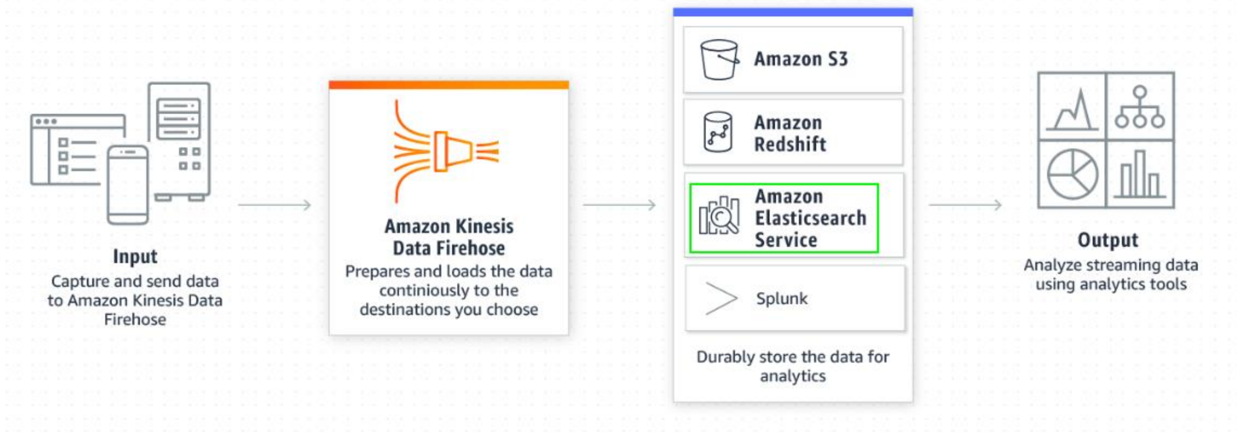

Input
Capture and send data to Amazon Kinesis Data Firehose
Amazon Kinesis Data Firehose
Prepares and loads the data continiously to the destinations you choose
Amazon S3
Amazon Redshift
Amazon Elasticsearch Service
Splunk
Durably store the data for analytics
Output
Analyze streaming data using analytics tools

Figure 2. Amazon Kinesis solution

Amazon Kinesis Data Firehose is the easiest way to reliably load data into data warehouses and analysis tools. The service is capable of importing, transforming and loading streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service and Splunk, enabling near real-time analytics with the business intelligence tools and dashboards you currently use. It is a fully managed service that automatically scales according to data throughput and does not require constant management. The service is also capable of batching, compressing, transforming and encrypting data before it is downloaded, minimizing the amount of memory used at the destination and increasing security [2].

Using Firehose, you do not need to create an application or manage resources. Configure the data producer – CloudWatch – to send data to Firehose and it will automatically send the data to the specified destination (see fig. 3).

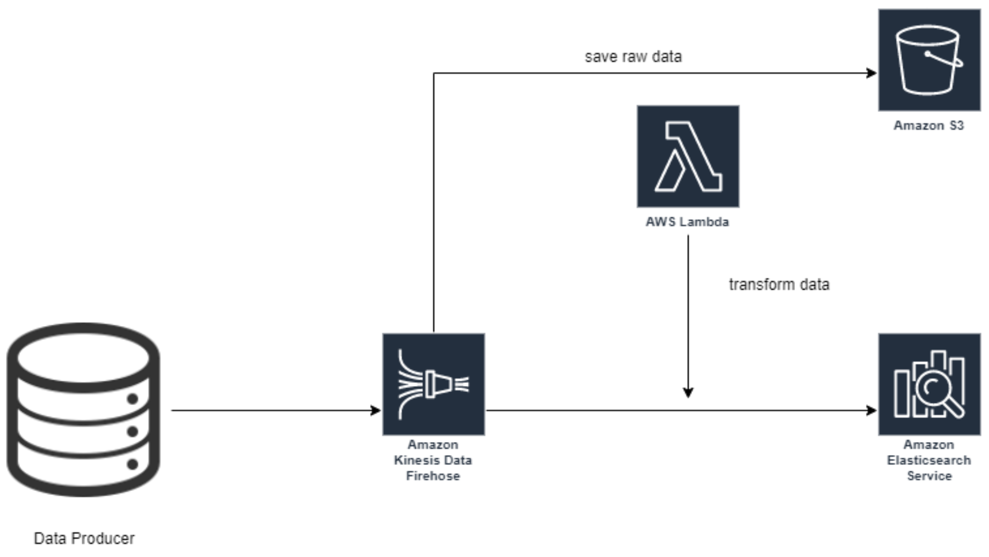


Figure 3. Amazon Kinesis Data Firehose approach

According to the above scheme, before sending the data to the ElasticSearch service, the Lambda function converts our data according to our requirements. Then the transformed records will be sent to ElasticSearch service via Kinesis Firehose. Meanwhile, our raw data will be sent to the AWS S3 bucket before conversion. There, using lifecycle rules, we can move them to AWS Glacier or other S3 categories [4].

Here is the example of using AWS Lambda to transform the Amazon Kinesis Firehose:

```
from __future__ import print_function
import base64
import msgpack
import json
print('Loading function')


def lambda_handler(event, context):
  output = []

  for record in event['records']:
    payload = msgpack.unpackb(base64.b64decode(record['data']), raw=False)

    # Do custom processing on the payload here
    output_record = {
        'recordId': record['recordId'],
        'result': 'Ok',
        'data': json.dumps(payload, ensure_ascii=False).encode('utf8')
    }
    output.append(output_record)

  print('Successfully processed {} records.'.format(len(event['records'])))
  return {'records': output}
```

Elasticsearch is a distributed RESTful search and analytics engine that allows you to search and analyze log data in real time. It centrally stores log data, allowing you to use it to draw key insights and improve long-term analytics.

Below is a final approach (see fig. 4) of how Kinesis data flow from the various CloudWatch rules to the destination, which in our case is Elasticsearch (fig. 4.) [6-7].

1) Kinesis receives events data from CloudWatch.

2) The data stream invokes a Lambda function which takes out request and response and decompresses the incoming log if it was compressed from the origin (in our case for simplicity's sake, we won't compress it) and puts them in an S3 bucket and further proceeds to replace the same with URL's which the Kibana users can click on later to view the req/res in their browsers.

3) Lambda then returns the transformed log back to Kinesis which then puts them into the AWS Elasticsearch service.

4) In case any logs fail and all retries fail (retries are configurable), the failed logs are put in a configurable S3 bucket which can then be processed seperately, note that you can also dump every log into an S3 bucket along with the Elasticsearch service, however unless you mind not getting a fat bill from AWS I would dissuade against that.

5) Logs can then be viewed by the user using Kibana.

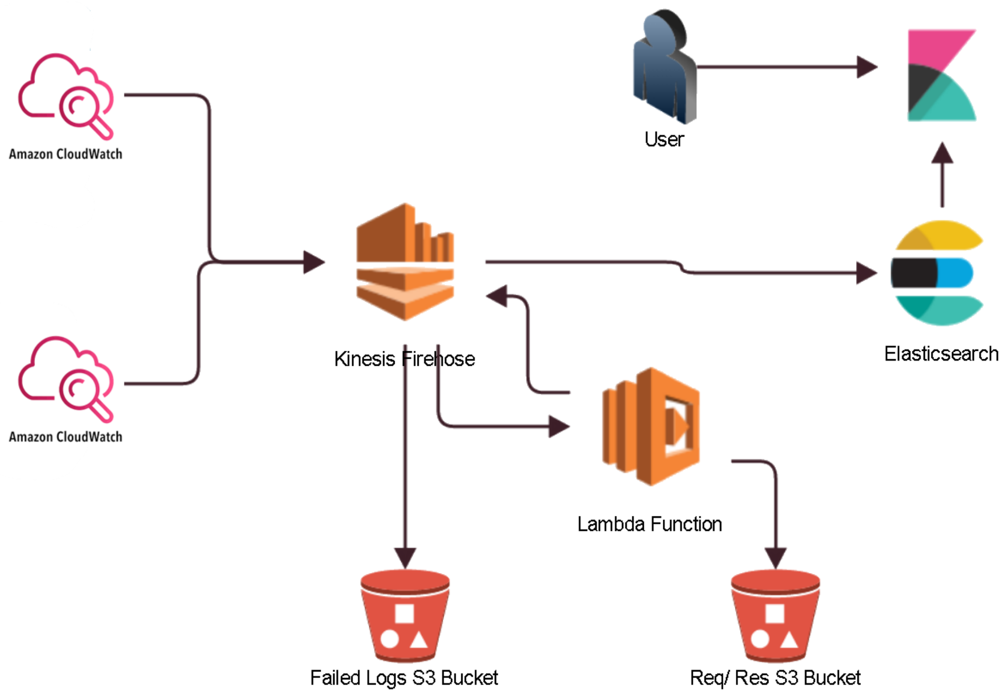


Figure 4. Our final solution for AWS Resources monitoring and analytics

In the below diagram (see fig. 5) displaying near real-time solution to monitor various EC2 events such as starting and shutting down instances, creating VPCs, etc. Likewise, we can build a continuous monitoring solution for all the API operations that are relevant to our daily AWS operations and resources [6].

The following charts demonstrates the API operations and the number of times that they have been triggered in the past 12 hours.

Figure 5. Discover and view the events.

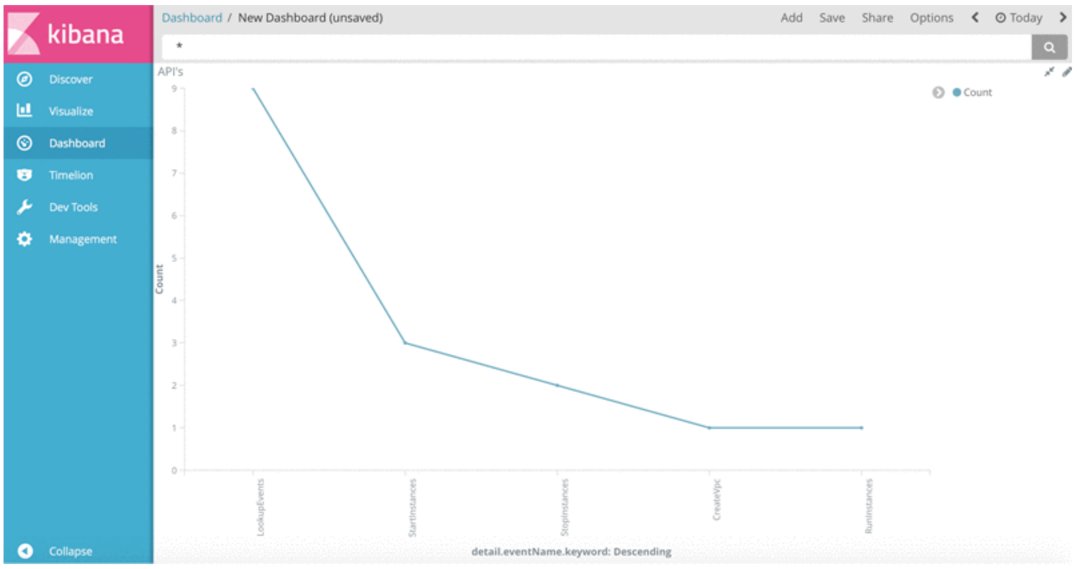


Figure 5. API operations and the number of times that they have been triggered

In short, in this AWS Amazon Web Services Real-Time AWS Resources Monitoring and Analytics solutions, we used a number of services like Amazon Kinesis Firehose, AWS Lambda functions, Amazon Elasticsearch, Amazon S3, AWS IAM Identity and Access Management service, Kibana as visualization and reporting tool and finally Amazon CloudWatch to collect AWS services events.

References

[1] Amazon AWS SDK for Python // [Online Resource] – Access Mode: https://aws.amazon.com/sdk-for-python// Access Date: 10.03.2021.

[2] Amazon Kinesis // [Online Resource] – Access Mode: https://aws.amazon.com/kinesis/?nc=sn&loc=1// Access Date: 02.23.2021.

[3] Monitoring the Amazon Kinesis Data Streams Service with Amazon CloudWatch// [Online Resource] – Access Mode: https://docs.aws.amazon.com/streams/latest/dev/monitoring-with-cloudwatch.html// Access Date: 03.07.2021.

[4] Using AWS Lambda with Amazon Kinesis Data Firehose// [Online Resource] – Access Mode: https://docs.aws.amazon.com/lambda/latest/dg/services-kinesisfirehose.html// Access Date: 03.04.2021.

[5] Firehose Boto3// [Online Resource] – Access Mode: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/firehose.html// Access Date: 03.04.2021.

[6] Kibana Guide – Access // [Online Resource] – Access Mode: https://www.elastic.co/guide/en/kibana/current/access.html// Access Date: 03.04.2021.

[7] Hybrid Integration Platform// [Online Resource] – Access Mode: https://www.elastic.io/Access Date: 03.04.2021.

[8] Cloudwatch logs subscription consumer// [Online Resource] – Access Mode: https://github.com/amazon-archives/cloudwatch-logs-subscription-consumer/tree/master/configuration/kibana/Access Date: 03.04.2021.

[9] Build a log Analytics Solution// [Online Resource] – Access Mode: https://aws.amazon.com/getting-started/hands-on/build-log-analytics-solution//Access Date: 03.04.2021.

[10] Use AWS Lambda with other services //[Online Resource] – Access Mode: https://docs.aws.amazon.com/lambda/latest/dg/lambda-services.html//Access Date: 03.04.2021.

МОНИТОРИНГ И АНАЛИТИКА AWS RESOURCES В РЕАЛЬНОМ ВРЕМЕНИ

К. С. ДИК
Аспирант БГУИР, инженер по программному обеспечению и данным

И.И. ПИЛЕЦКИЙ
к.м.н., доцент кафедры информатики БГУИР.

Белорусский государственный университет информатики и радиоэлектроники
E-mail: kanstantind@gmail.com, ianmenski@gmail.com

Аннотация В этой статье описывается подход к мониторингу и аналитике ресурсов AWS в реальном времени. Основным является централизация журналов, событий и показателей для облачных приложений, работающих на Amazon Web Services (AWS). В этой работе представлен подход, как можно использовать вместе следующие сервисы AWS: Amazon CloudWatch, CloudTrail, Kinesis, Elasticsearch, Lambda, Simple Cloud Storage, EC2.

Ключевые слова: мониторинг, аналитика, облако, AWS, CloudWatch, Kinesis, Elasticsearch, Lambda, S3, EC2.