Author: Madhurima Rawat

Implementing Cloud Monitoring and Logging

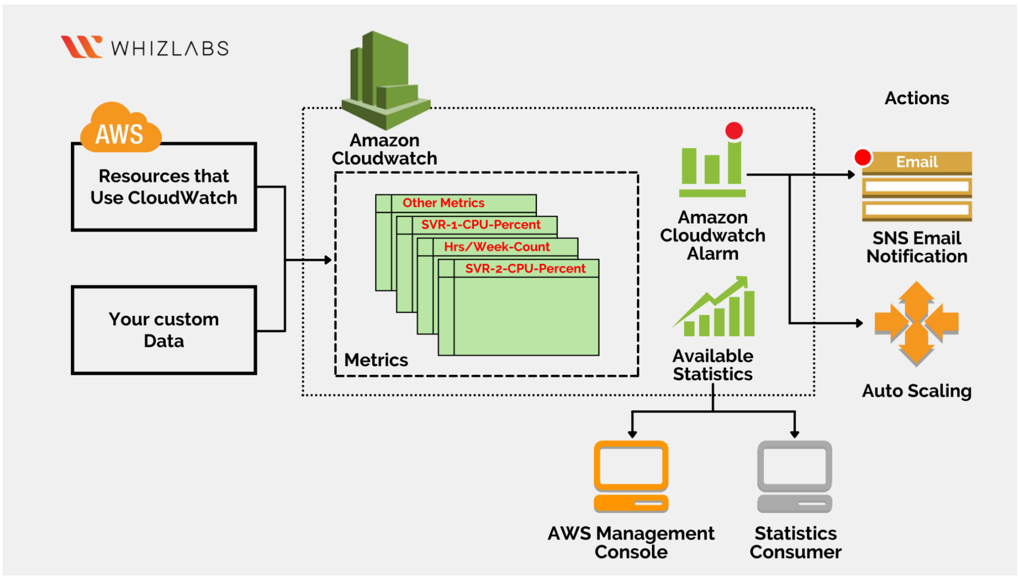
This experiment simulates AWS CloudWatch logging using Docker, LocalStack, and AWS CLI. It involves setting up log groups, streams, and sending log events locally. This enables monitoring and testing without an actual AWS account.

Introduction

This experiment simulates **AWS CloudWatch logging** using **Docker, LocalStack, and AWS CLI**. It allows setting up **log groups, log streams, and sending log events** locally without needing an **actual AWS account**. This setup helps in testing and monitoring log data before deploying it to a real AWS environment.

What is AWS CloudWatch?

AWS CloudWatch is a **monitoring and observability** service that collects and tracks **metrics, logs, and events** to provide insights into cloud resources and applications.

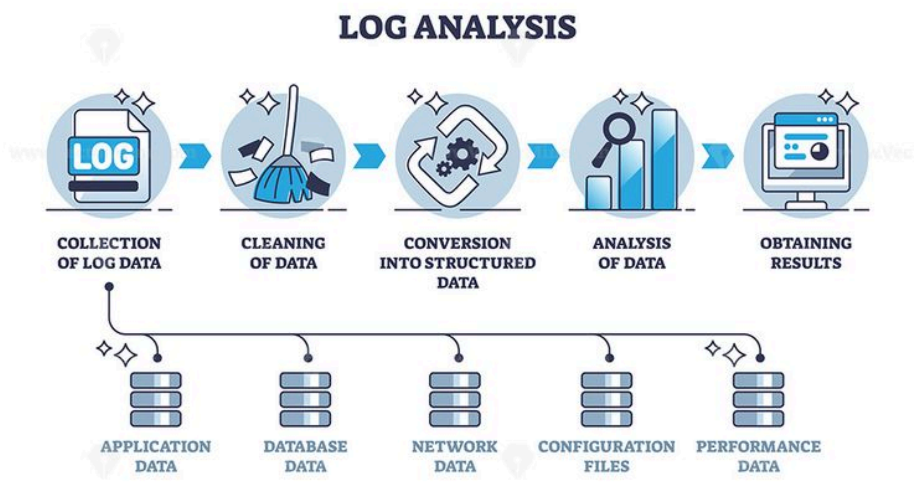


🔹 Key Features of CloudWatch:

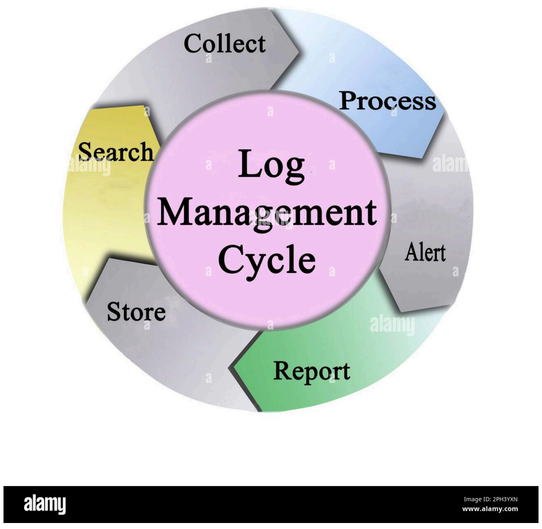
- **Metrics Monitoring** 📊: Collects and analyzes performance metrics for AWS services.
- **Log Management** 📜: Stores and processes logs for applications and infrastructure.
- **Alarms & Notifications** 🚨: Triggers alerts based on threshold values.
- **Dashboards & Visualization** 📈: Provides real-time monitoring dashboards.
- **Event-driven Automation** ⚙️: Automates actions based on events.

Understanding Log Management

Log management is a crucial part of **cloud monitoring** and involves the collection, storage, and analysis of logs.



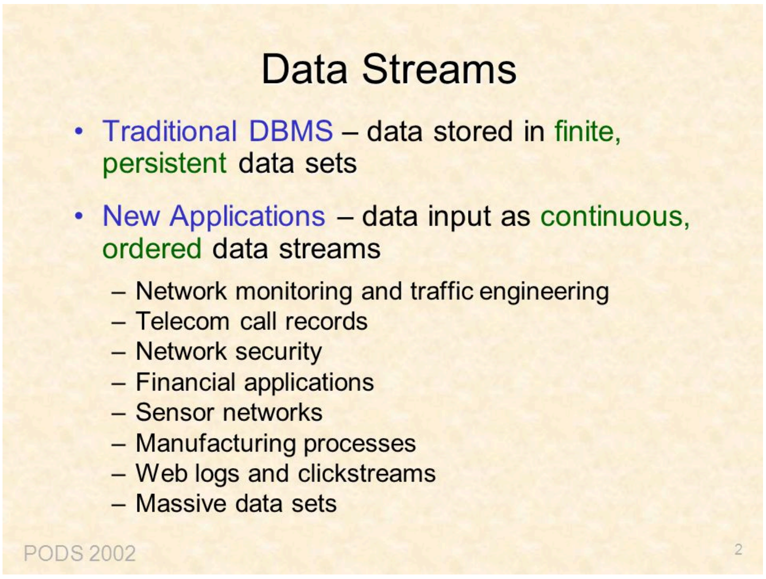
💡 Six Components of Log Management:



1. **Log Collection** 🏗️: Logs are gathered from various sources (servers, applications, cloud services).
2. **Log Aggregation** 🗃️: Logs are combined and stored in a centralized system like CloudWatch Logs.
3. **Log Storage** 🏢: Logs are kept in CloudWatch **Log Groups** and **Log Streams**.
4. **Log Analysis** 🔍: Tools like AWS CloudWatch Insights analyze log data for patterns and trends.
5. **Log Monitoring & Alerts** 🚨: CloudWatch can send alerts based on predefined conditions.
6. **Compliance & Retention** ✅: Logs are archived based on security and compliance requirements.

What Are Log Streams?

Log streams represent a sequence of **log events** from the same source (e.g., an application instance, EC2 instance, or microservice).



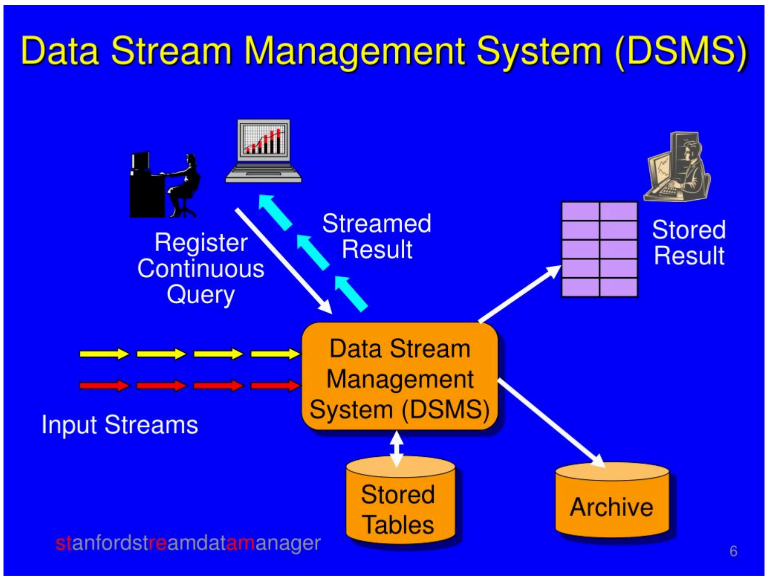
Data Streams

- Traditional DBMS – data stored in finite, persistent data sets
- New Applications – data input as continuous, ordered data streams
  - – Network monitoring and traffic engineering
  - – Telecom call records
  - – Network security
  - – Financial applications
  - – Sensor networks
  - – Manufacturing processes
  - – Web logs and clickstreams
  - – Massive data sets

PODS 2002 2

◆ **Example:**

- A web server might have separate log streams for **each instance** it runs on.
- CloudWatch logs are organized in **log groups**, and each log group contains multiple **log streams**.



Data Stream Management System (DSMS)

Register Continuous Query
Streamed Result
Stored Result
Data Stream Management System (DSMS)
Input Streams
Stored Tables
Archive
stanfordstreamdatamanager
6

Real-life Use Case: CloudWatch Logs & Metrics

Scenario:
A company hosts its web application on AWS **EC2 instances** and wants to **monitor CPU usage** and **analyze logs** for errors.

Steps in AWS CloudWatch:

1️⃣ **Enable CloudWatch Logs** on EC2 instances to store application logs.
2️⃣ **Create a log group** `/my/app/logs` and **define log streams** for different instances.

3️⃣ **Send logs & metrics** (e.g., CPU usage, errors, response time) to CloudWatch.
4️⃣ **Set up alarms** to notify the DevOps team when CPU usage exceeds **80%**.
5️⃣ **Analyze logs** using CloudWatch Insights to detect anomalies.

Conclusion

This experiment demonstrates how **CloudWatch logging and monitoring** work in a local environment using **Docker, LocalStack, and AWS CLI**. It helps in **log management, real-time monitoring, and alerting** before deploying applications to AWS.

AWS CLI Commands for LocalStack (CloudWatch and Logs)

1. Create a Log Group

```
aws --endpoint-url=http://localhost:4566 logs create-log-group
--log-group-name /my/app/logs
```

Description:

- Creates a log group named `/my/app/logs` in AWS CloudWatch Logs.

2. Create a Log Stream

```
aws --endpoint-url=http://localhost:4566 logs create-log-stream
--log-group-name /my/app/logs --log-stream-name my-stream
```

Description:

- Creates a log stream named `my-stream` within the `/my/app/logs` log group.

3. Put Log Events

```
aws --endpoint-url=http://localhost:4566 logs put-log-events --
log-group-name /my/app/logs --log-stream-name my-stream --log-
events "[{\"timestamp\":1741348140000,\"message\":\"Test log entry\"}]"
```

Description:

- Adds a log entry with a timestamp (`1741348140000`) and message (`Test log entry`) to the `my-stream` log stream.

Output:

```
-----------------------------------------------------------------------------------
|                                  PutLogEvents                                   |
+--------------------+--------------------------------------------------------------+
|  nextSequenceToken |  00000000000000000000000000000000000000000000000000000001  |
+--------------------+--------------------------------------------------------------+
||                              rejectedLogEventsInfo                             ||
|+------------------------------------------------------------------+-----------+|
||  tooNewLogEventStartIndex                                        |  0        ||
|+------------------------------------------------------------------+-----------+|
```

4. Put Metric Data

```
aws --endpoint-url=http://localhost:4566 cloudwatch put-metric-
data --namespace "MyApp" --metric-name "CPUUsage" --value 75
```

Description:

- Publishes a custom CloudWatch metric `CPUUsage` with a value of `75` in the `MyApp` namespace.

5. List CloudWatch Metrics

```
aws --endpoint-url=http://localhost:4566 cloudwatch
list-metrics --namespace "MyApp"
```

Description:

- Lists all CloudWatch metrics under the `MyApp` namespace.

Output:

```
-------------------------------
|         ListMetrics         |
+-----------------------------+
||          Metrics          ||
|+-------------+-------------+|
|| MetricName  |  Namespace  ||
|+-------------+-------------+|
||  CPUUsage    |  MyApp      ||
|+-------------+-------------+|
```

6. Describe Log Groups

```
aws --endpoint-url=http://localhost:4566 logs describe-log-groups
```

Description:

- Retrieves details of all log groups available in CloudWatch Logs.

Output:

```
-------------------------------------------------------------------------------------------
|                                   DescribeLogGroups                                     |
+-----------------------------------------------------------------------------------------+
||                                      logGroups                                        ||
|+------------------+--------------------------------------------------------------------+|
||  arn             |  arn:aws:logs:us-east-1:000000000000:log-group:/my/app/logs:*      ||
||  creationTime    |  1741322075726                                                     ||
||  logGroupName    |  /my/app/logs                                                      ||
||  metricFilterCount|  0                                                                ||
||  storedBytes     |  0                                                                 ||
|+------------------+--------------------------------------------------------------------+|
```

7. Retrieve Log Events

```
aws --endpoint-url=http://localhost:4566 logs get-log-events
--log-group-name /my/app/logs --log-stream-name my-stream
```

Description:

- Fetches the log events from `my-stream` under `/my/app/logs` .

Output:

```
-------------------------------------------------------------------------------------
|                                  GetLogEvents                                     |
+--------------------+--------------------------------------------------------------+
|  nextBackwardToken |  b/00000000000000000000000000000000000000000000000000000000  |
|  nextForwardToken  |  f/00000000000000000000000000000000000000000000000000000000  |
+--------------------+--------------------------------------------------------------+
```