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.

This document provides a comprehensive breakdown of all commands, inputs, outputs, and their explanations, ensuring a clear understanding of each step in the workflow.

Command 1: Creating a Log Group and Log Stream

Command:

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

Explanation:

- `aws logs create-log-group` → Creates a new **log group** in AWS CloudWatch.
- `--log-group-name /my/app/logs` → Specifies the name of the **log group**.
- `--endpoint-url=http://localhost:4566` → Uses **LocalStack** to simulate AWS services locally.

Command:

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

Explanation:

- `aws logs create-log-stream` → Creates a **log stream** within an existing log group.
- `--log-group-name /my/app/logs` → Specifies the log group where the stream will be created.
- `--log-stream-name my-stream` → Defines the **log stream name**.
- `--endpoint-url=http://localhost:4566` → Uses **LocalStack**.

Command 2: Inserting Log Events

Command (Incorrect):

```
aws --endpoint-url=http://localhost:4566 logs put-log-events --
log-group-name /my/app/logs --log-stream-name my-stream --log-
events timestamp=$(date +%s%3N),message="Test log entry"
```

Error:

```
Error parsing parameter '--log-events': Expected: '=',
received: '+' for input:
{self._error_location()}
```

Cause:

- The `timestamp=$(date +%s%3N)` syntax is **not supported** in Windows `cmd` .
- The AWS CLI expects a **proper JSON object** for `--log-events` .

Corrected Command (Windows PowerShell):

```
$timestamp = [int]((Get-Date -UFormat %s) * 1000)
aws --endpoint-url=http://localhost:4566 logs put-log-events --
log-group-name /my/app/logs --log-stream-name my-stream --log-
events "[{\"timestamp\":$timestamp,\"message\":\"Test log entry\"}]"
```

Explanation:

- `$timestamp = [int]((Get-Date -UFormat %s) * 1000)` → Generates a **Unix timestamp in milliseconds**.
- `--log-events "[{\"timestamp\":$timestamp,\"message\":\"Test log entry\"}]"` → Sends a JSON-formatted log entry.
- `--endpoint-url=http://localhost:4566` → Uses **LocalStack**.

Command 3: Troubleshooting Errors

Error:

```
Error parsing parameter '--log-events': Invalid JSON:
Expecting value: line 1 column 15 (char 14)
```

Cause:

- The command contains an **unresolved variable** (`$(Get-Date -UFormat %s)000`), which results in invalid JSON.

Fix:

Use PowerShell to properly resolve variables before executing the command.

Error:

```
'$timestamp' is not recognized as an internal or external
command, operable program or batch file.
```

Cause:

- `$timestamp` is a **PowerShell variable**, but the command is being run in **cmd**.
- `cmd` does not support PowerShell syntax.

Fix:

Run the command in **PowerShell**, not `cmd` . If using `cmd` , manually replace `$timestamp` with a **hardcoded** Unix timestamp.

Command 4

```
for /f %i in ('powershell -Command "Get-Date -UFormat %%s"')
do set timestamp=%i000
```

Explanation:

- This command retrieves the current Unix timestamp (in seconds) using PowerShell.
- The `-UFormat %s` option formats the date as a Unix timestamp.
- The `for /f` loop captures this output and assigns it to the variable `timestamp` , multiplying it by 1000 to convert it into milliseconds.

Potential Issue:

- The variable `%i` is correctly set within the loop, but the assignment might not be persisting in the batch script.

Command 5

```
set timestamp=%s000
```

Explanation:

- This sets `timestamp` to `%s000`, but `%s` has not been defined properly.
- Instead of setting `timestamp` to an actual numeric value, it's being assigned a literal `%s000`.

Error:

- When this value is used in JSON formatting later, it results in an invalid JSON string with an undefined timestamp.

Command 6

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

Explanation:

- This command attempts to send log events to AWS LocalStack's CloudWatch logs.
- The JSON format requires `timestamp` to be a valid numeric value.
- The value `%timestamp%` is not correctly substituted, leading to an invalid JSON structure.

Error:

```
Error parsing parameter '--log-events': Invalid JSON:
Expecting value: line 1 column 15 (char 14)
```

Cause:

- `%timestamp%` is still `%s000`, causing JSON parsing to fail.

Command 7

```
for /f %i in ('powershell -Command "Get-Date -UFormat %%s"')
do @set timestamp=%i000
```

Explanation:

- The `@` symbol prevents the command from being echoed in the terminal.
- This properly assigns `timestamp` in milliseconds.

Fix:

- Use `echo %timestamp%` after running this to confirm the value is properly set.

Command 8

```
aws --endpoint-url=http://localhost:4566 logs put-log-events
--log-group-name /my/app/logs --log-stream-name my-stream
--log-events "%logEvents%"
```

Explanation:

- Attempts to pass a JSON log entry stored in `logEvents` .
- The JSON formatting is incorrect, leading to an error.

Error:

```
Invalid JSON: [{timestamp:%s000,message:Test
```

Cause:

- `%s000` was not substituted properly.
- Missing double quotes around `timestamp` value in JSON.

Command 9

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

Error:

```
Invalid JSON: Expecting value: line 1 column 15 (char 14)
JSON received: [{"timestamp":%s000,"message":"Test log entry"}]
```

Cause:

- `%timestamp%` still contains `%s000`, not an actual number.

Fix:

- Ensure `%timestamp%` is properly set using:

  ```
  for /f %i in ('powershell -Command "Get-Date -UFormat %%s"')
  do set timestamp=%i000
  ```

Command 10

```
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\"}]"
```

Explanation:

- Here, the timestamp `1741348140000` is manually set.
- This is a valid Unix timestamp in milliseconds.

Success Output:

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

Explanation:

- The log was successfully submitted, but it was rejected because the timestamp might be too far in the future.

Command 11: Put a Metric in CloudWatch

Command:

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

Explanation:

This command sends a **custom metric** named **CPUUsage** with a value of **75** to **CloudWatch** under the **namespace "MyApp"**.

Expected Output:

🚫 **No output**
Since this command only submits the metric, AWS CLI does not return any confirmation message.

Command 12: List Metrics in CloudWatch

Command:

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

Explanation:

This command retrieves and displays all the **metrics** available under the namespace `"MyApp"`.

Output Breakdown:

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

- 🔹 **ListMetrics**: This section shows the retrieved metrics.
- 🔹 **Metrics Table**:

- **MetricName**: `"CPUUsage"` → This is the name of the metric we added in Step 1.
- **Namespace**: `"MyApp"` → This confirms that the metric belongs to the correct namespace.

👉 This confirms that the metric was successfully added and is now stored in CloudWatch.

Command 13: Describe Log Groups in CloudWatch Logs

Command:

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

Explanation:

This command lists all **log groups** present in AWS CloudWatch Logs.

Output Breakdown:

```
-------------------------------------------------------------------------------------
|                                  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                                                           ||
|+-------------------+-------------------------------------------------------------+|
```

- 🔹 **logGroups**: This section displays information about available log groups.
- 🔹 **ARN (Amazon Resource Name)**:

- `"arn:aws:logs:us-east-1:000000000000:log-group:/my/app/logs:*"` → The unique identifier for the log group in AWS.
  🔹 **creationTime**: `"1741322075726"` (Epoch timestamp) → Represents when the log group was created.
  🔹 **logGroupName**: `"/my/app/logs"` → Name of the log group that stores log events.
  🔹 **metricFilterCount**: `"0"` → No metric filters are currently applied to this log group.
  🔹 **storedBytes**: `"0"` → No log data is stored yet.

👉 This confirms that a log group named **"/my/app/logs"** exists but currently does not contain any log data.

Command 14: Retrieve Log Events from a Log Stream

Command:

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

Explanation:

This command fetches **log events** from the log stream **"my-stream"** inside the log group **"/my/app/logs"**.

Output Breakdown:

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

🔹 **nextBackwardToken**:

- `"b/00000000000000000000000000000000000000000000000000000000"`
- This token is used for **navigating backward** (older log entries).

🔹 **nextForwardToken**:

- `"f/00000000000000000000000000000000000000000000000000000000"`
- This token is used for **navigating forward** (newer log entries).

📌 **Why is there no actual log data?**

- The output only contains navigation tokens because **no logs have been written** to this stream yet.
- If logs were present, they would be displayed in the output.

Summary of Outputs

Command	What It Does	Key Output Details
`put-metric-data`	Adds a CloudWatch metric	✅ No output, but metric is added
`list-metrics`	Lists metrics in CloudWatch	✅ Shows `"CPUUsage"` under `"MyApp"` namespace

Command	What It Does	Key Output Details
`describe-log-groups`	Lists available log groups	✅ Shows `/my/app/logs` but no logs yet
`get-log-events`	Retrieves log events	⚠️ No actual logs found, only navigation tokens