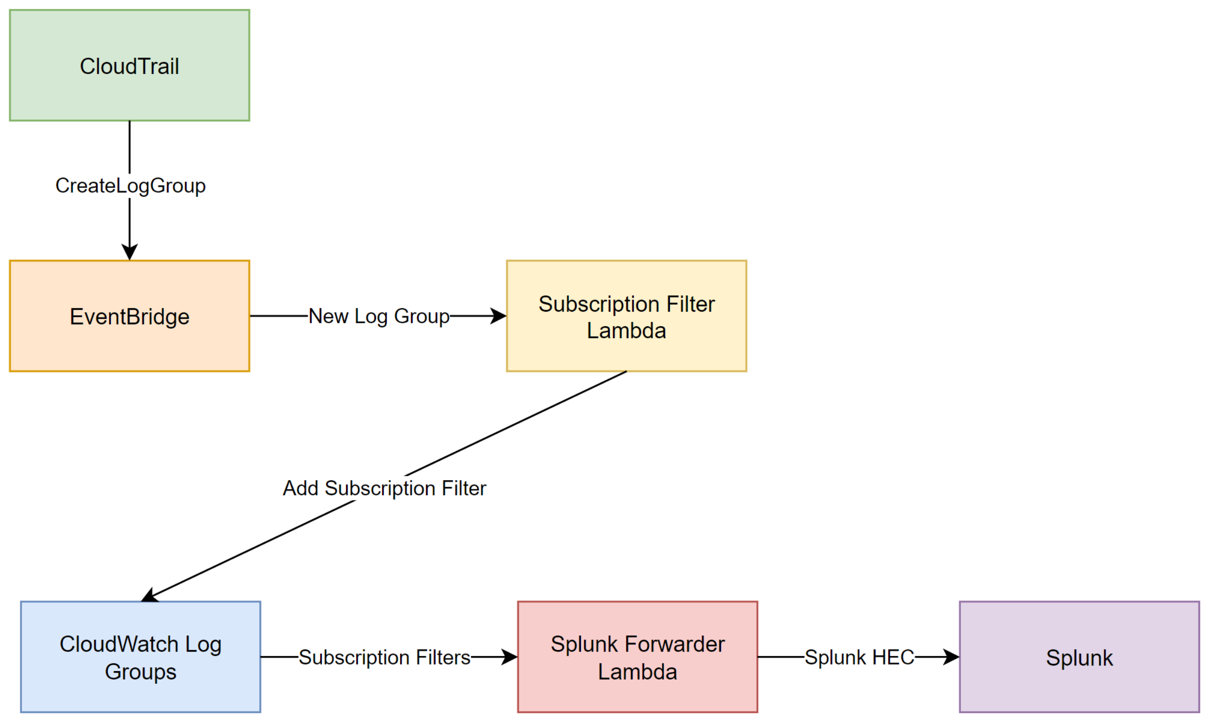
CloudWatch Logs to Splunk Solution Design

Introduction

This design document outlines the architecture and components of the solution devised to forward logs from AWS CloudWatch to Splunk. The primary objective is to ensure that all logs generated by AWS Lambdas, which are automatically captured in CloudWatch, are seamlessly and efficiently sent to Splunk for centralized logging, monitoring, and analysis. Leveraging AWS Lambda functions and Splunk's HTTP Event Collector (HEC), this solution aims to provide a reliable, scalable, and secure mechanism for log aggregation and forwarding.

System Overview

High-level Diagram:



Data Flow 1

CloudTrail:

- CloudTrail detects CreateLogGroup API calls and places an event on the default event bus.

EventBridge:

- Upon new log group creation, an EventBridge rule matches on CreateLogGroup management events from CloudTrail and triggers the SubscriptionFilterHandler lambda.

SubscriptionFilterHandler Lambda:

- Conducts a systematic audit across all existing CloudWatch Log Groups.
- Verifies and enforces the existence of a subscription filter on each log group, ensuring that logs are directed towards the Splunk Forwarder Lambda.
- This ensures that any new or pre-existing log group without a subscription filter is attended to, maintaining a uniform policy for log processing.
- This lambda doesn't require a specific input and thus can be triggered from the AWS Lambda Console to perform an audit of all log groups at any time.

Add Subscription Filter:

- An operation executed by the Subscription Filter Lambda.
- It involves the attachment of subscription filters to CloudWatch Log Groups that are missing them, thereby enabling the forwarding of logs to the SplunkForwarder lambda.

Data Flow 2

CloudWatch Logs:

- Source of all the log data for AWS Lambdas.
- Every AWS Lambda function execution, error, or other related logs are automatically sent to CloudWatch Logs.

Subscription Filters:

- Located within CloudWatch Logs Groups.
- Act as a mechanism to forward logs from all Lambda log groups without any filtering.
- As new logs are generated from the Lambda executions, they are automatically matched by the subscription filters and forwarded to the Splunk Forwarder Lambda.

Splunk Forwarder Lambda:

- AWS Lambda function specifically designed to process and forward the incoming Lambda log data.
- Once it receives the logs from CloudWatch, the function safely parses, prepares, and possibly enriches the logs for forwarding.
- Logs produced by the StructuredLogging library will automatically have their metadata indexed to the enriched logs sent to Splunk.
- It then utilizes the Splunk's HTTP Event Collector (HEC) endpoint to send the logs to Splunk efficiently.

Splunk HEC:

- HTTP Event Collector endpoint provided by Splunk.
- Specially configured to receive the Lambda log data over HTTP(s).
- Ensures secure and efficient data ingestion into Splunk from the Lambda environment.

Splunk:

- The destination where all Lambda logs are indexed and stored.
- Upon ingestion through the HEC, logs are immediately available for searching, alerting, visualization, and detailed analysis within Splunk.

Subscription Filter Handler Lambda

Operational Flow:

Log Group Enumeration:

- On invocation, the function enumerates all CloudWatch Log Groups, specifically targeting those related to AWS Lambda functions (identified by the prefix `/aws/lambda/`).
- It excludes log groups associated with the Splunk Forwarder Lambda itself to avoid self-logging scenarios.

Subscription Filter Management:

- The function operates in two modes, governed by the `REMOVE_SUBSCRIPTIONS` environment variable.
- In the addition mode (`REMOVE_SUBSCRIPTIONS` = `"false"` ***NOTE: String not Boolean**), the function attaches subscription filters to log groups that are missing them.
- In the removal mode (`REMOVE_SUBSCRIPTIONS` = `"true"` ***NOTE: String not Boolean**), the function removes existing subscription filters from the log groups.
- This is intended as a "rollback" mechanism in case of unforeseen issues, eliminating the need to manually remove subscription filters from every log group, and to remove subscription filters from log groups in lower environments where Splunk log forwarding may not be desired.

Direct Application of Subscription Filter Changes:

- For each log group processed, the function directly applies the appropriate action—either adding or removing a subscription filter—based on its current configuration and the mode of operation.
- This active management ensures that new and existing log groups are consistently configured to forward logs to the Splunk Forwarder Lambda.
- Triggering this lambda from the AWS Lambda Console for manual auditing of log groups is supported.

Error Handling and Retry Mechanism:

- The function incorporates error handling for scenarios such as API rate limiting and AWS service-related exceptions.
- A retry mechanism with exponential backoff is implemented to handle `ThrottlingException`, enhancing the function's robustness under varying API request volumes.
- The error handling was created in a way that should not interrupt the operation of the lambda in the case of an error.

Logging and Monitoring:

- The solution utilizes the StructuredLogging library for structured, detailed logging of operations and errors.
- This approach allows for efficient troubleshooting in case of errors, such as the existence of two subscription filters in a log group, which prevents the addition of a new one.

Reserved Concurrency:

- The lambda function has been assigned a reserved concurrency of **one**.
- This is to ensure that multiple instances of the lambda function don't trip each other up.

- In the case that multiple log groups are created in a short time frame, any concurrent executions will wait until any previous execution finishes before running.
- For this reason, the lambda was assigned the maximum timeout duration of 15 minutes, as concurrent executions will not wait longer than the previous execution timeout duration.

Splunk Forwarder Lambda

Operational Flow:

Log Event Processing:

- Upon invocation, the function receives an event from CloudWatch Logs. This event contains log data in a compressed and encoded format.
- The function first decodes and decompresses the log data using the `prepareLogEvents` function.

Log Data Transformation and Enrichment:

- Each log event is processed and transformed into a suitable format for ingestion by Splunk.
- Additional metadata, such as the log group name and environment variables, are added to the log data to enrich it and provide more context in Splunk.
- Fields to be considered as important metadata that accompanies each forwarded log message:
    - System Name
    - Component Name
    - Tracking ID
    - Environment Name
    - Human readable timestamp

Splunk Index Token Retrieval:

- The Splunk Index Token is securely retrieved from AWS Secrets Manager, ensuring sensitive data is not exposed or hardcoded in the function.
- The function includes robust error handling to manage scenarios where the token retrieval fails, such as network issues or access permission problems, ensuring the function's resilience.
- Caching of the retrieved token has been implemented to optimize performance. This minimizes the number of calls to AWS Secrets Manager, reducing latency and cost.

Sending Logs to Splunk:

- The function iterates over the processed log events and sends them to Splunk using the HEC endpoint.
- For each log event, a POST request is made to Splunk with the necessary headers and body, including the authorization token and the transformed log data.

Error Handling and Resilience:

- The function includes error handling to manage scenarios such as network failures, issues with the Splunk endpoint or problems retrieving the index token from secrets manager.
- In case of a failure in transmitting a log event to Splunk, an error is logged for further investigation without halting the processing of further logs.
- The error handling has been designed to only throw exceptions in the case of a failure when retrieving the Splunk index token, as this would render the lambda unfunctional.

Monitoring and Feedback:

- The function logs the outcome of each attempt to send logs to Splunk, providing visibility into the success or failure of log forwarding.
- These logs can be monitored to ensure the reliable operation of the log forwarding process and to facilitate troubleshooting in case of issues.

Reserved Concurrency:

- The lambda function has been assigned a reserved concurrency of **ten**.
- This should ensure an instance of the lambda can be spun up at any time without impacting the concurrency pool limit of the relevant AWS account.

Scalability and Performance:

Potential Bottlenecks:

- **Lambda Throttling**: While Lambda is designed to scale, we're aware of potential throttling if the concurrency limits are reached. Proper monitoring should be in place to alert us to any such occurrences.
- **Log Volume**: As log volume increases, the Lambda might need to process more batches. This has been taken into account in its design to ensure smooth processing.

Splunk Ingestion Limits:

- **Splunk HTTP Event Collector (HEC)**: When data is sent to Splunk via the HEC, there's a maximum limit of 1 million bytes (or 1 megabyte) for each event or batch of events. This is a fixed limit and cannot be modified.
- **Splunk Port 9997**: When data is sent to Splunk via port 9997 (typically used for forwarding data from Splunk Universal Forwarders), the size limit is different. If an event exceeds 10,000 bytes, it will be truncated, and only the first 10,000 bytes of the event will be ingested by Splunk.

Deployment Process Using Terraspace

1. **Pre-requisites**:
    - `Splunk Forwarder` lambda must be built at least once. It would need to be rebuilt if there are code changes.
    - Necessary AWS permissions to apply Terraform configurations.
    - Access to the Splunk console for verifying logs.
    - Ensure all environment variables like `SPLUNK_INDEX_TOKEN`, `SPLUNK_REQUEST_CHANNEL`, and `SPLUNK_URL` are set appropriately.
2. **Deployment**:
    - Navigate to the `<rootDir>/src/lambda/splunk-forwarder` directory.
    - Run `npm run build-win`.
    - Navigate to the `<rootDir>/src/lambda/subscription-filter-handler` directory.
    - Run `npm run build-win`.
    - Return to `<rootDir>` and run `bundle exec terraspace up logging_service.`
3. **Destroy** (if needed to remove the infrastructure):
    - Navigate to `<rootDir>` (The project root directory with the Terrafile)
    - Run `bundle exec terraspace down logging_service.`
4. **Post-deployment Tests or Checks**:
    - Manually create a new CloudWatch log group to trigger The SubscriptionFilterHandler lambda.
    - Check the logs of the SubscriptionFilterHandler lambda to ensure correct operation.
    - Go to https://nswdoe.splunkcloud.com/en-GB/app/search/search?q=search%20index%3Ddoe_digital_integration
    - Select a timeframe from the dropdown to the right of the search bar.
    - Logs from CloudWatch should begin to appear momentarily.
    - In the case that logs are not appearing after a few minutes, check the logs of the SplunkForwarder lambda to identify any issues with log transmission.

blocked URL

Deployment Process using Jenkins

Please consult Logging Service Splunk Forwarder Deployment (Under Reconstruction) - Digital Integration - ICC - Confluence (nsw.gov.au) for instructions on deploying from CI/CD pipeline.

Security Considerations

1. **Secrets Management**:
    - The Splunk Index token is provided to the Lambda function by Secrets Manager. Please make this available in the relevant AWS environment as outlined in Logging Service Splunk Forwarder Deployment (Under Reconstruction) - Digital Integration - ICC - Confluence (nsw.gov.au).
2. **Encryption**:
    - While our logging policy mandates that no sensitive data should be logged, thereby mitigating some risks, always ensure that no sensitive data inadvertently ends up in the logs.
    - Encryption at rest or in-transit is not deemed necessary given the non-sensitive nature of the logs based on our policy. However, it's worth noting that AWS Lambda encrypts function data at rest and automatically encrypts it when the function is running.
3. **VPC Considerations**:
    - The Lambda function does not interact directly with other AWS services in this setup, which simplifies the networking considerations and eliminates the requirement of the lambda running inside a VPC.
4. **IAM Roles/Policies and Network ACLs**:
    - The Lambda function's IAM role has been provided with only the permissions it needs and nothing more (Principle of Least Privilege).

Maintenance and Future Enhancements

Known Limitations:

1. **Latency:** Depending on the complexity of the Lambda function and the volume of logs processed, there may be a slight delay in sending logs to Splunk.
2. **Log Volume:** High volumes of logs can result in throttling on the AWS Lambda side or potential bottlenecks in Splunk ingestion, especially if using the HTTP Event Collector (HEC).
3. **Log Retention:** The current setup might not handle long-term log storage. Over time, Splunk storage costs may increase if log retention policies aren't properly managed.

Potential Areas for Future Improvements:

1. **Enhanced Error Handling:** Improvements can be made to handle different error types more effectively, including retries for failed log transmissions.
2. **Log Filtering:** Implement advanced log filtering within the Lambda function to reduce unnecessary log transfers to Splunk.
3. **Data Enrichment:** Enhance the Lambda function to add more metadata or enrich logs before sending to Splunk for more insightful analytics.
4. **Multi-destination Support:** The ability to send logs to multiple destinations (different Splunk instances or other platforms) for better redundancy or specialized analysis.

Maintenance:

1. **Lambda Function Updates:** Ensure that the Lambda function is regularly updated for security patches and optimized for performance.
2. **Splunk Configuration Changes:** Regularly review and update Splunk inputs, index settings, and source type definitions to ensure optimal performance and accurate log parsing.
3. **Debugging Issues:** The solution has been designed to catch and log any errors it may encounter. To troubleshoot logs not appearing in Splunk, check the SplunkForwarder lambda logs. To troubleshoot the logs not making it to the SplunkForwarder lambda, check the SubscriptionFilterHandler lambda logs.