Prerequisites:<https://github.com/luckyeven/k8s-google-sso.git>、

# 

# Part 1: Understanding Anomaly Detection

Anomaly detection is a critical aspect of cybersecurity, helping to identify unusual patterns that may signify a security threat. Before integrating an anomaly detection system into our log analysis, a thorough understanding of various techniques and their applicability to security logs is crucial.

Anomaly detection techniques can be broadly categorized into the following types:

Statistical Methods: These methods assume that normal behavior follows a well-defined statistical distribution. Any significant deviation from this distribution signals an anomaly. Techniques include:

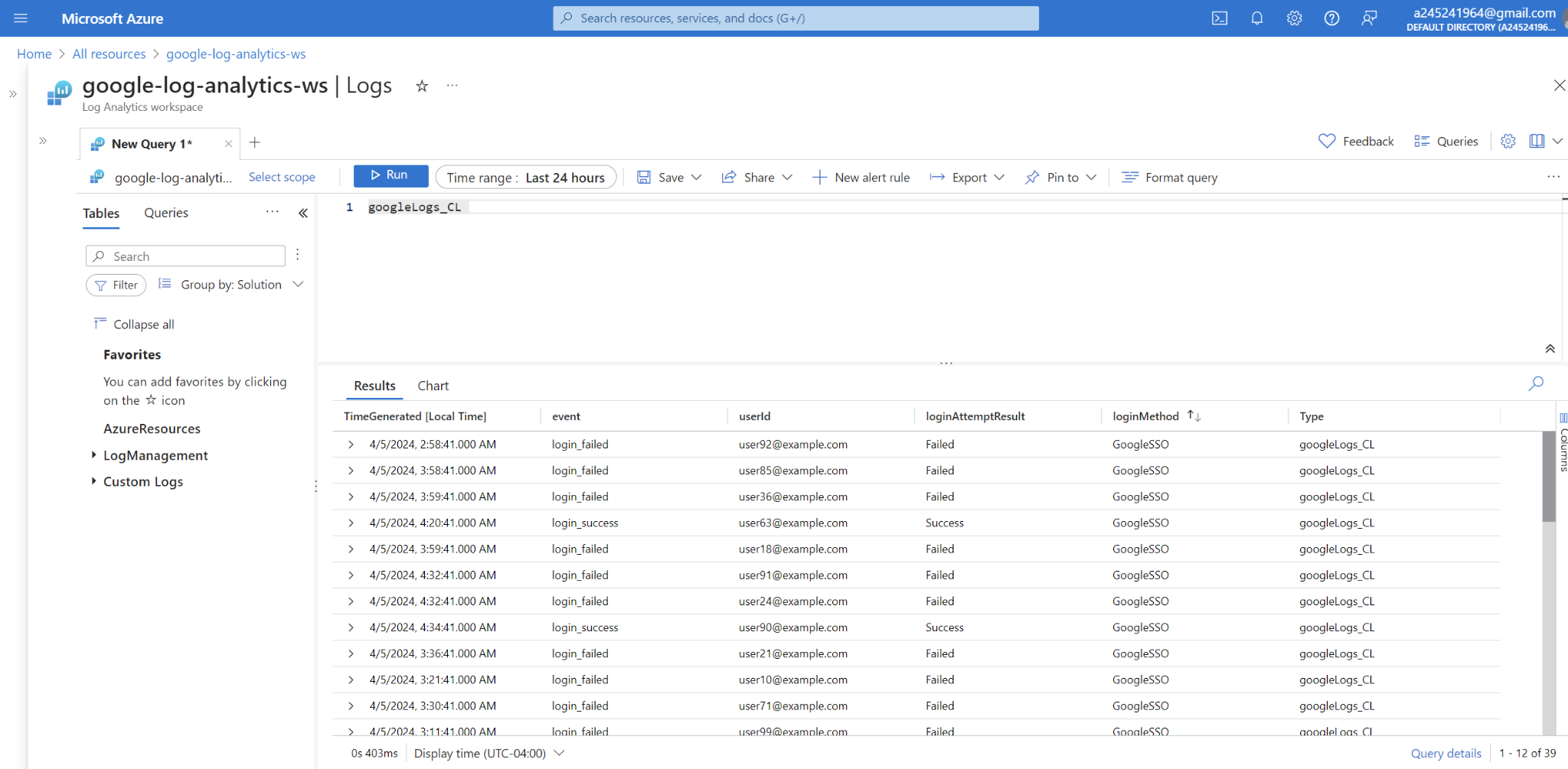
Supervised learning (e.g., using labeled data to train models on known anomalies)

Unsupervised learning (e.g., clustering algorithms like K-means or DBSCAN)

Semi-supervised learning (combining labeled and unlabeled data)

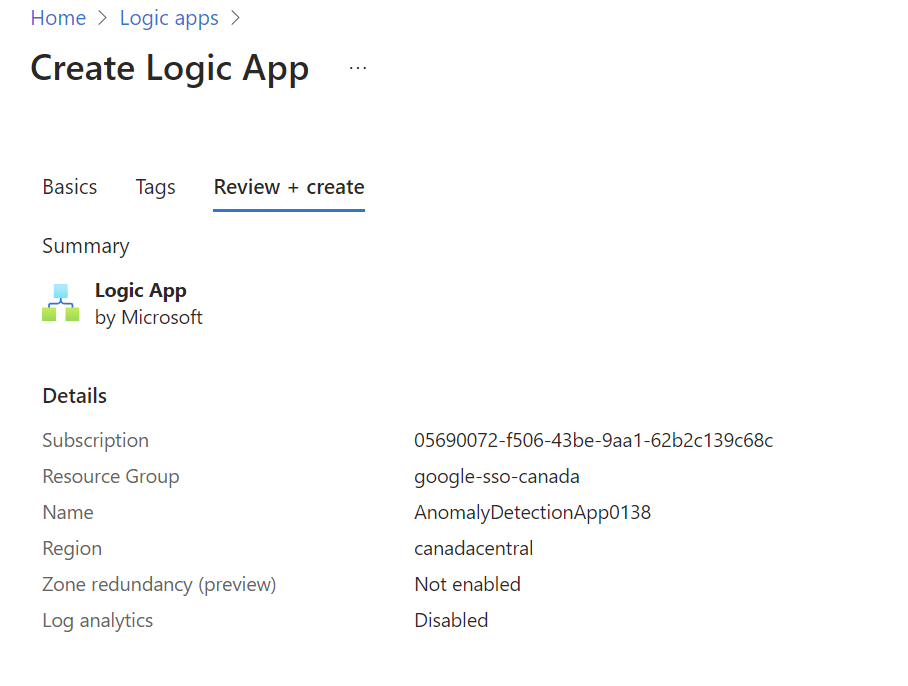
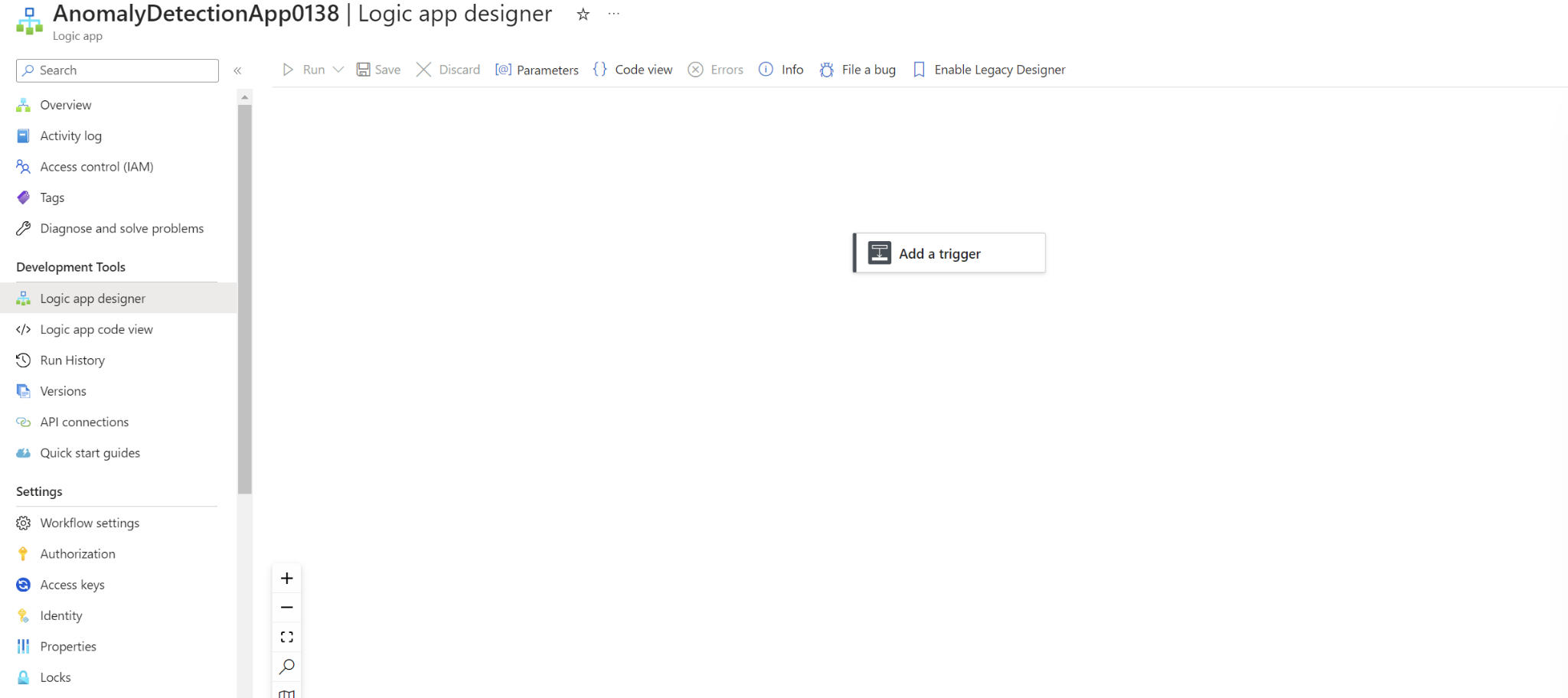
Rule-Based Methods: These involve setting explicit rules that define normal behavior. Anything that violates these rules is considered an anomaly. This method is straightforward but requires in-depth domain knowledge to set accurate rules.

for this assignment, we focus on Rule-Based Methods. we built on the code from the first assignment to log in using Google's single sign-on (SSO) feature. This process generates login logs, which we then send to Azure Log Analytics Workspace. The image below shows the logs created from our simulated login attempts. We'll specifically look at the "loginAttemptResult" to spot any unusual or suspicious login activity.

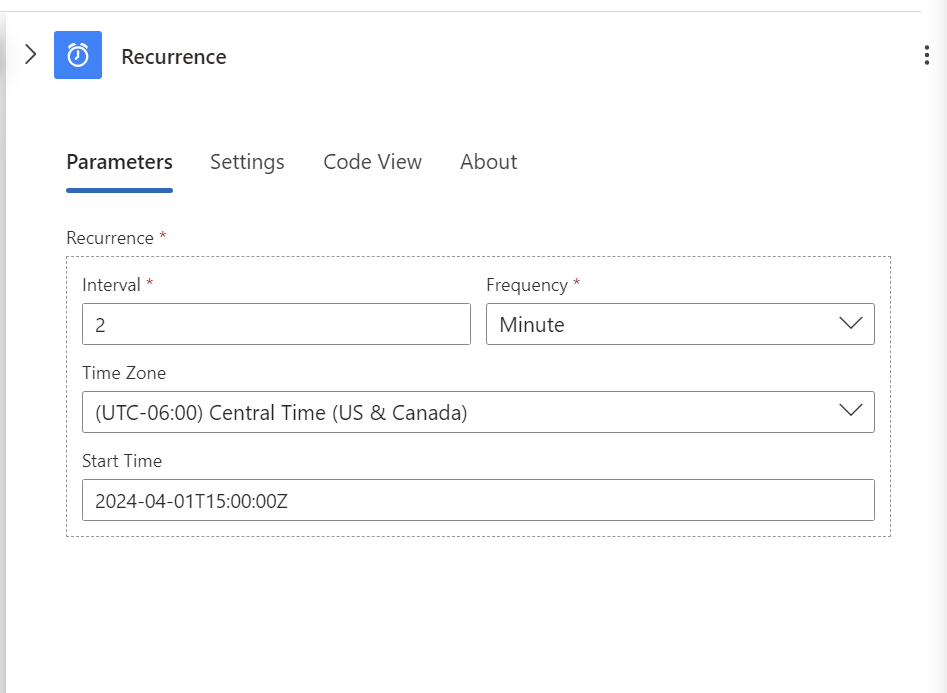
By analyzing the login attempt results, we can identify patterns that might indicate unauthorized access attempts or other security issues that need attention. This analysis helps strengthen our system’s security by quickly addressing any potential login anomalies detected in the logs.

# Part 2: Preparing for Automation

## Creating an Azure Logic App

Create a Logic App: Go to "Create a resource" > "Integration" > "Logic App"  
  
open the "Logic App Designer".  


## Setting Up Triggers and Actions

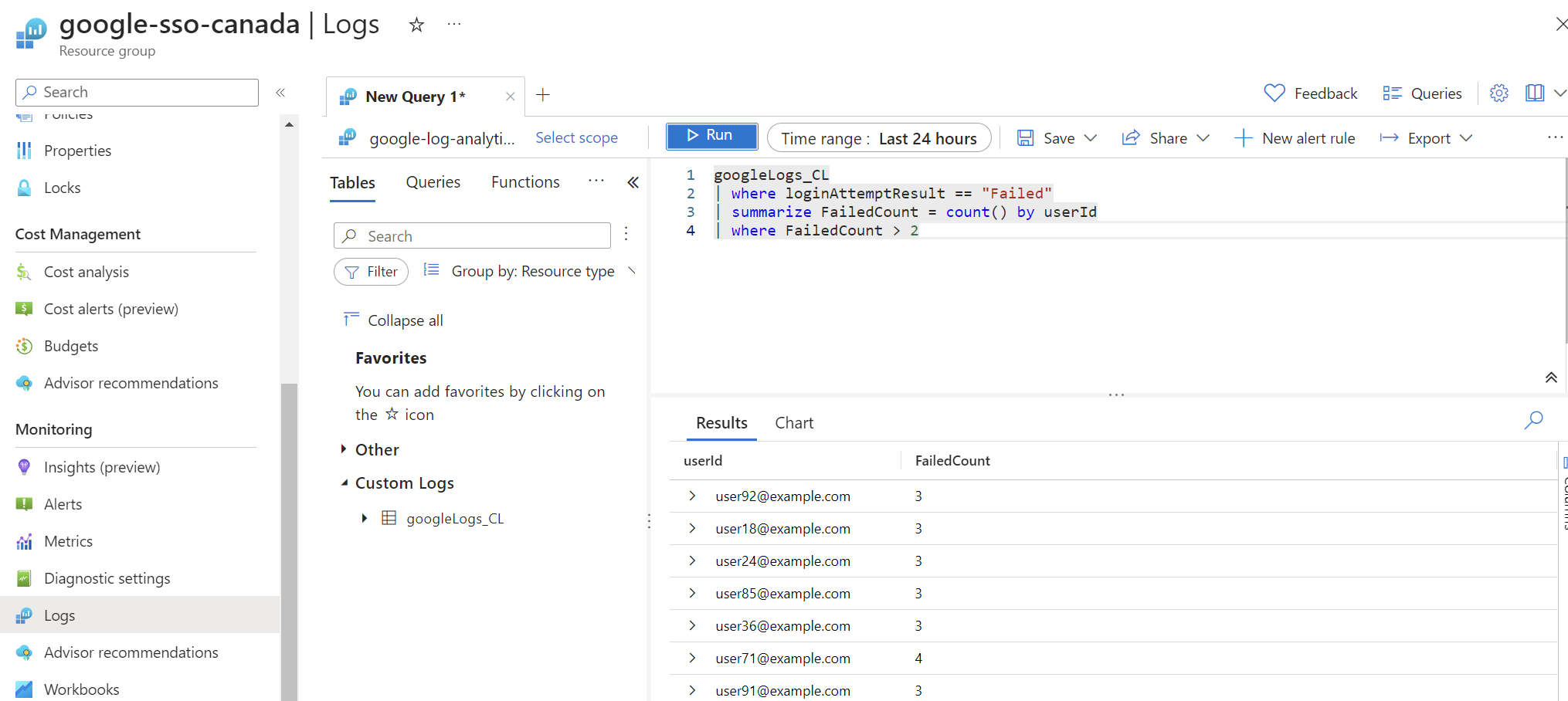
In Logic App configuration, set the trigger to activate every two minutes. This ensures app checks for new data or performs its designated tasks with this regular frequency, maintaining up-to-date operations.

After setting trigger, add an Action:

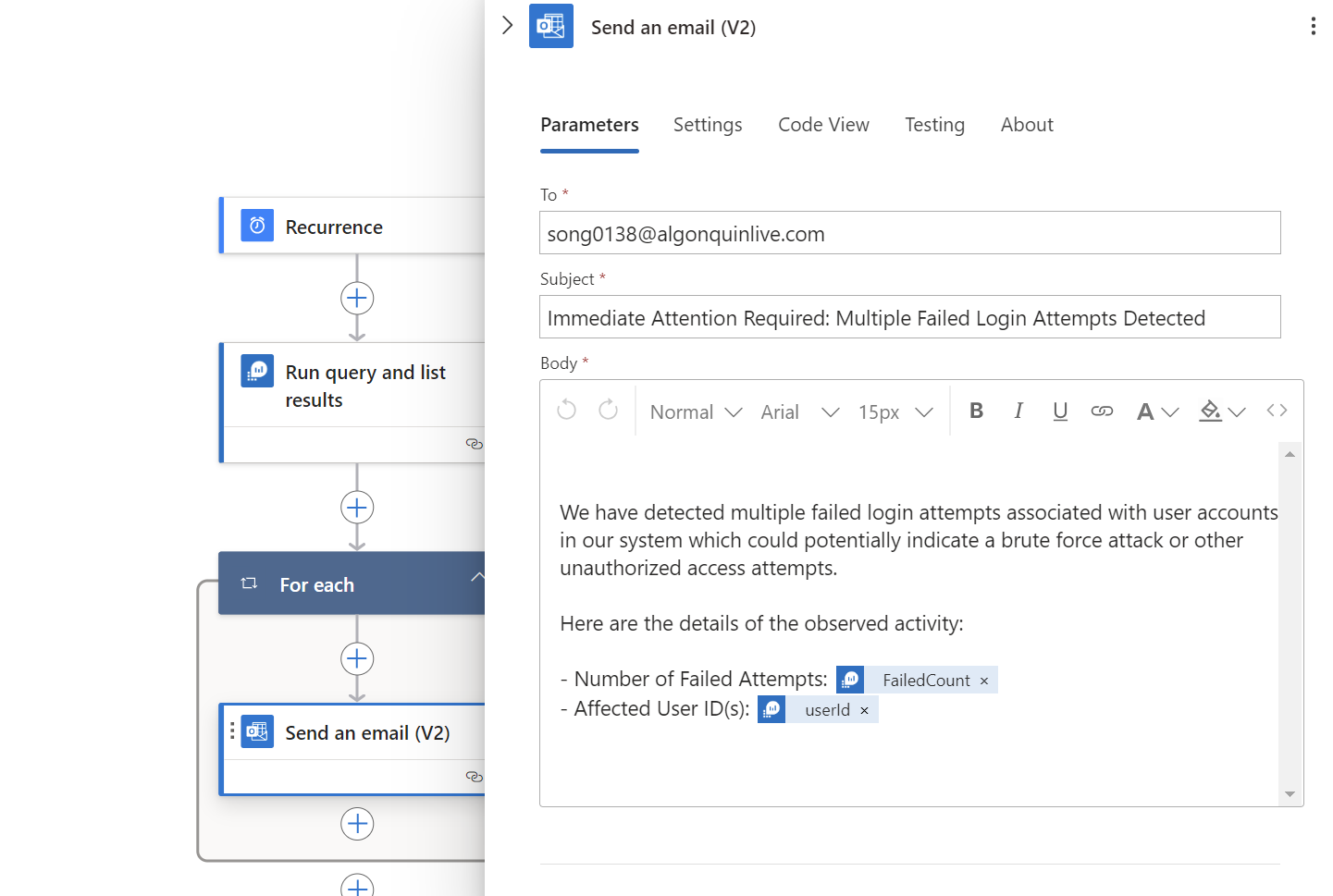
# Part 3: Implementing Anomaly Detection

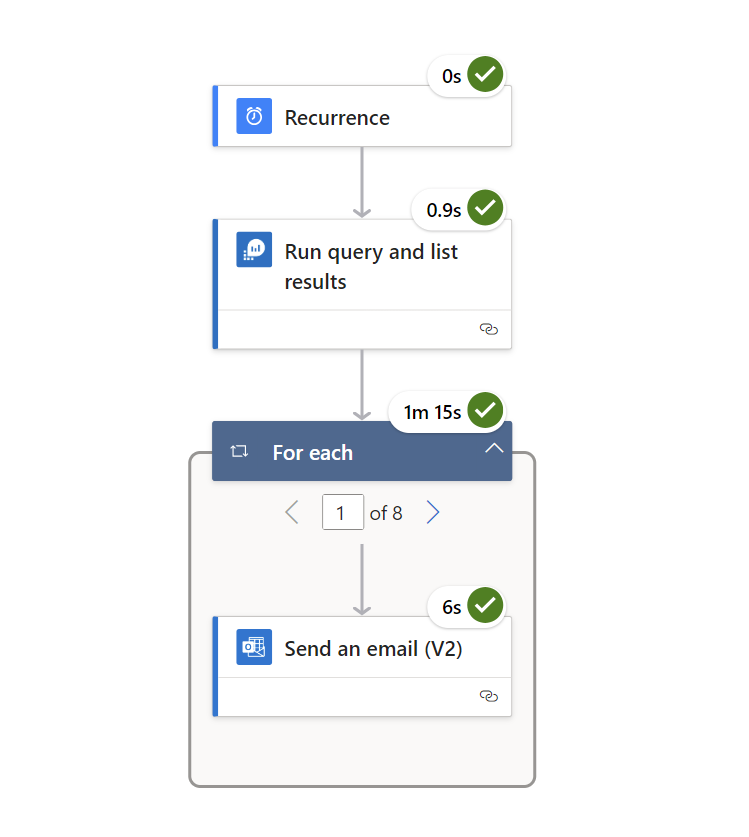
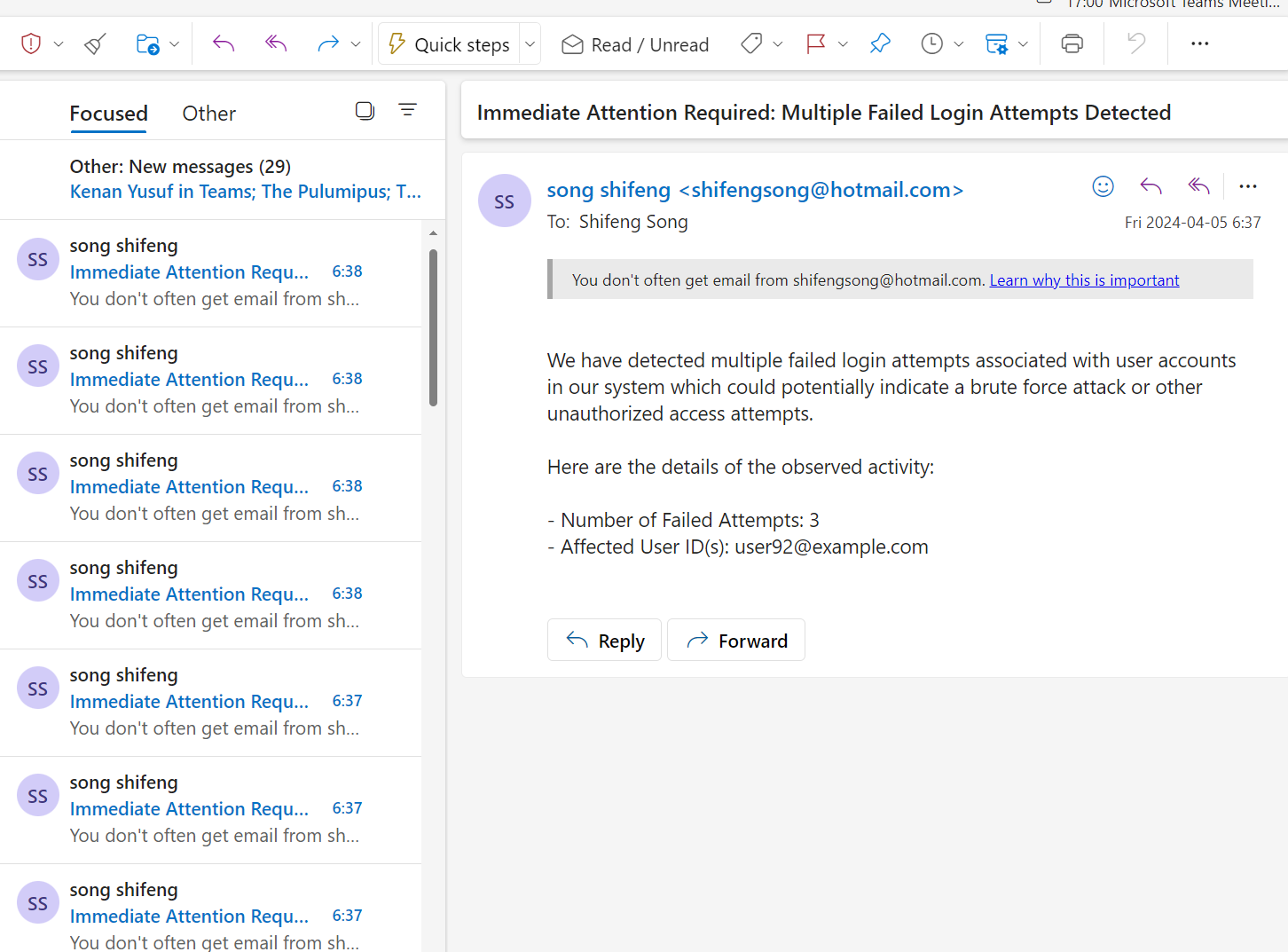
To construct a query targeting multiple failed login attempts for the same email address within the googleLogs\_CL table in the google-log-analytics-ws Log Analytics workspace, use Azure's Kusto Query Language (KQL). The KQL query would scan through the collected log data, specifically looking for patterns where a userId (email address) has experienced more than two failed login attempts.

The query structure would involve selecting from the googleLogs\_CL table where the loginAttemptResult equals "Failed". Next, you would aggregate the count of these failed attempts by userId. A final step would be to use a where filter to only include results where the FailedCount is greater than 2.

This query helps identify potential security risks, pinpointing accounts that might be under a brute force attack or users who are struggling with login, perhaps due to forgotten credentials. By automating the monitoring of such events, administrators can quickly take action, like initiating password resets or investigating possible attacks, thus maintaining the integrity and security of the system.

Next, by incorporating a new action in Logic App, to trigger an email notification whenever the query identifies more than two failed login attempts.



Upon executing the Logic App, an alert for unusual login attempts is successfully dispatched to the specified email addresses. This automation streamlines the monitoring process, ensuring prompt awareness and response to potential security incidents involving account access.  
  


# Part 4: Integrating Cloud Security Best Practices

Alert Escalation:

Imagine a scenario where an account experiences multiple login failures in a short period. Such patterns are often indicative of unusual activity, potentially signaling unauthorized access attempts. To enhance security measures and ensure prompt response to these incidents, the alerting mechanism within a system can be upgraded.

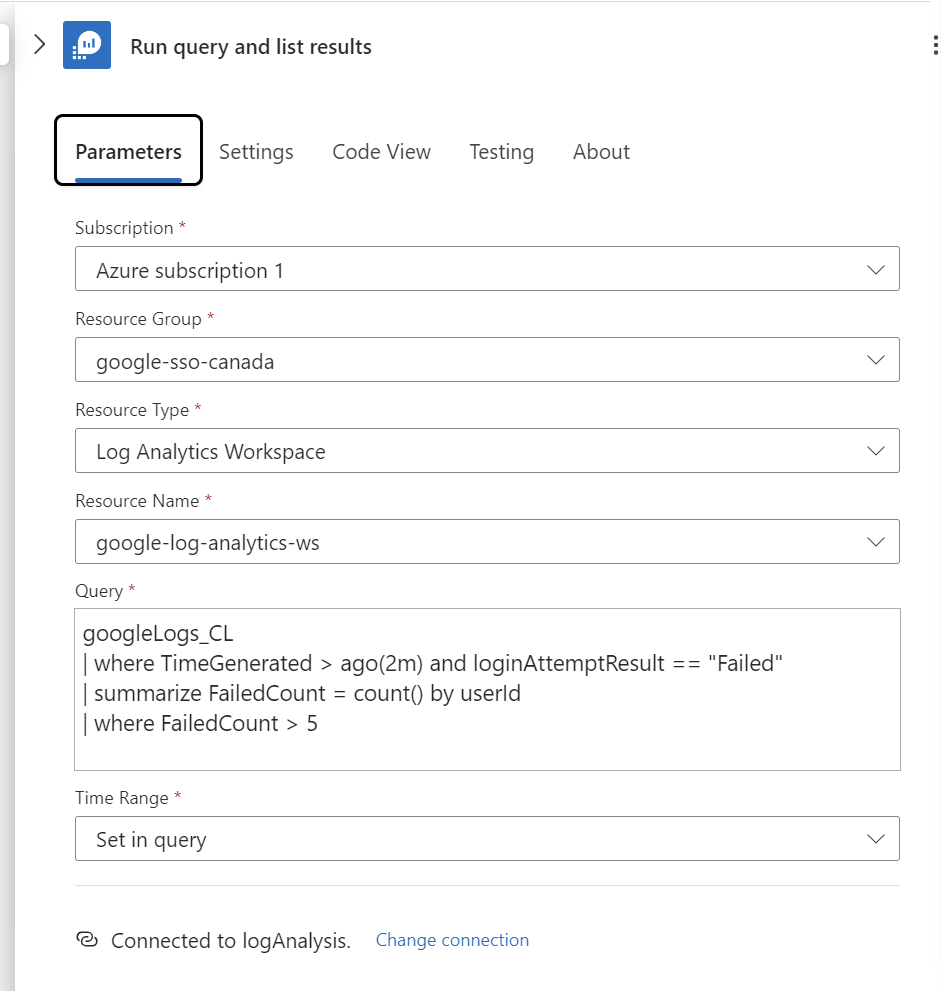
The upgraded alert system is designed to identify instances where an account logs in unsuccessfully more than five times within a two-minute window. This threshold is chosen based on the understanding that while users may occasionally mistype their passwords, exceeding this limit suggests something beyond typical user error.

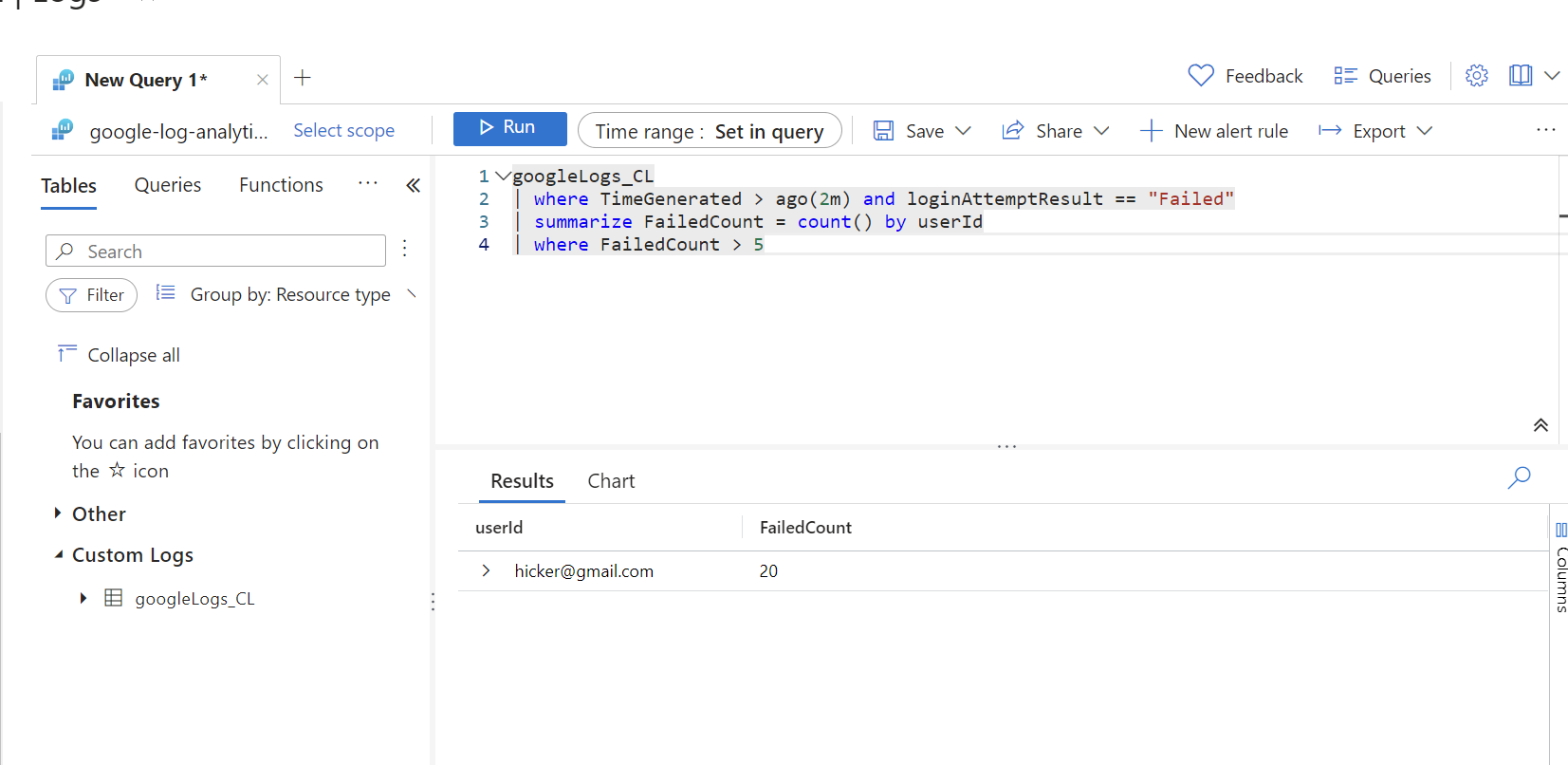
Upon detecting such behavior, the system triggers an automated process. This process involves compiling the relevant information about the suspicious activity, including the account details, timestamp of the occurrences, and any other pertinent data that could assist in a subsequent investigation.

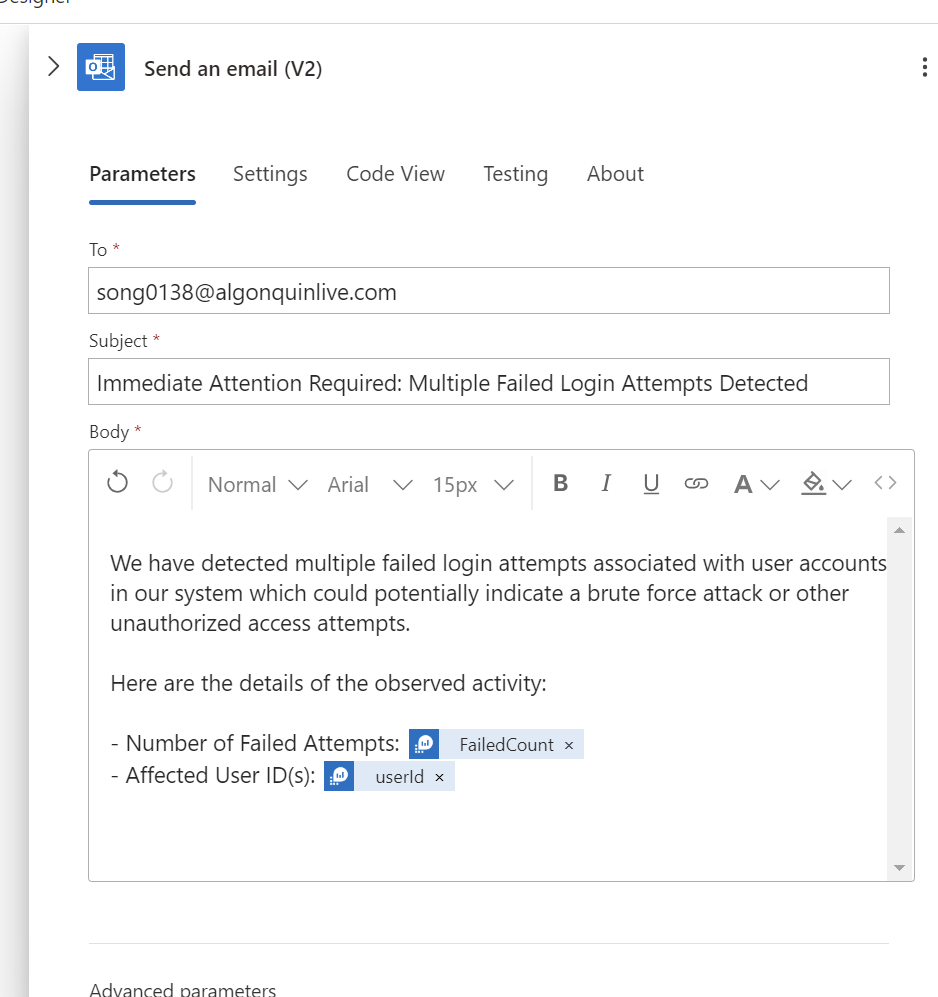
The culmination of this process is the dispatch of an email alert to a predetermined set of recipients. These recipients, typically members of the security or IT support teams, are entrusted with the task of reviewing the alert. They must assess the situation, determining whether the failed login attempts are indeed an anomaly requiring intervention or a benign incident that can be resolved with minimal action.

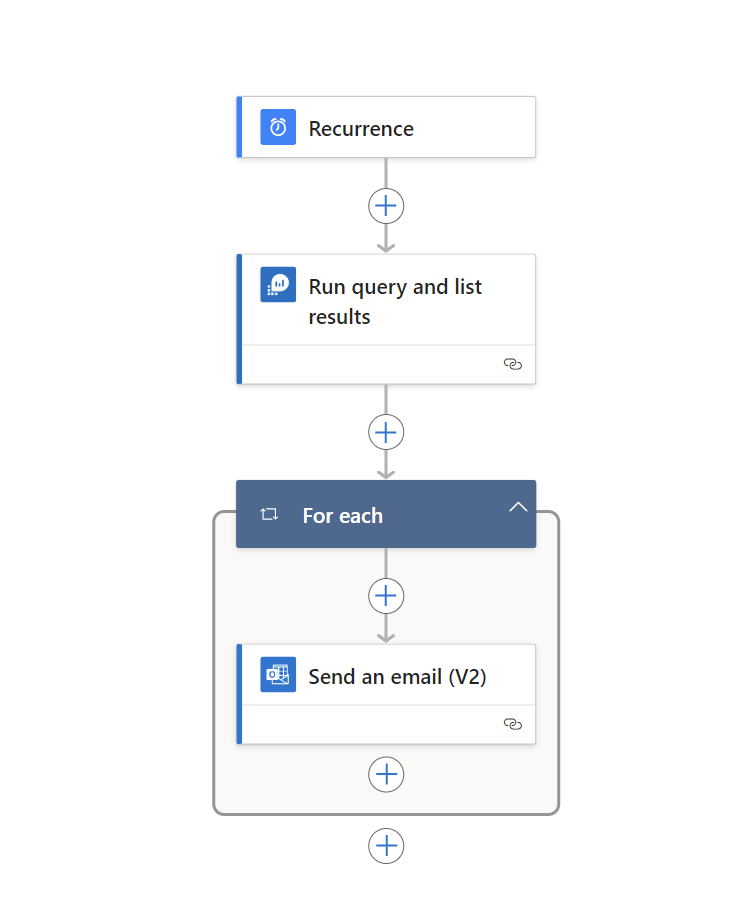
This proactive approach to monitoring and alerting ensures that potential security breaches are flagged and addressed promptly. It not only helps in safeguarding the integrity of the account but also plays a crucial role in maintaining the overall security posture of the organization. The ability to quickly identify and react to unusual login patterns is a testament to the system’s robustness against evolving cybersecurity threats.

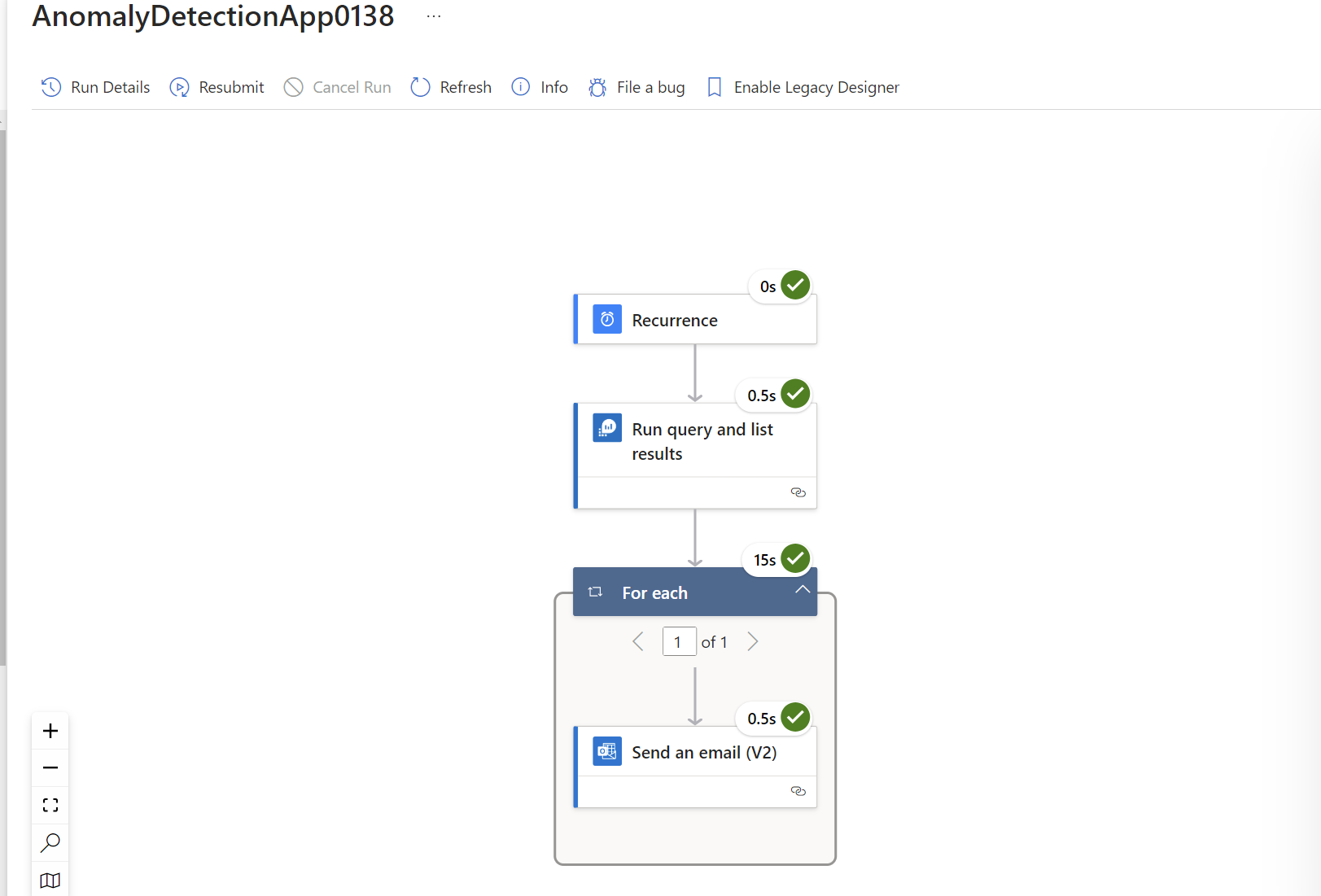
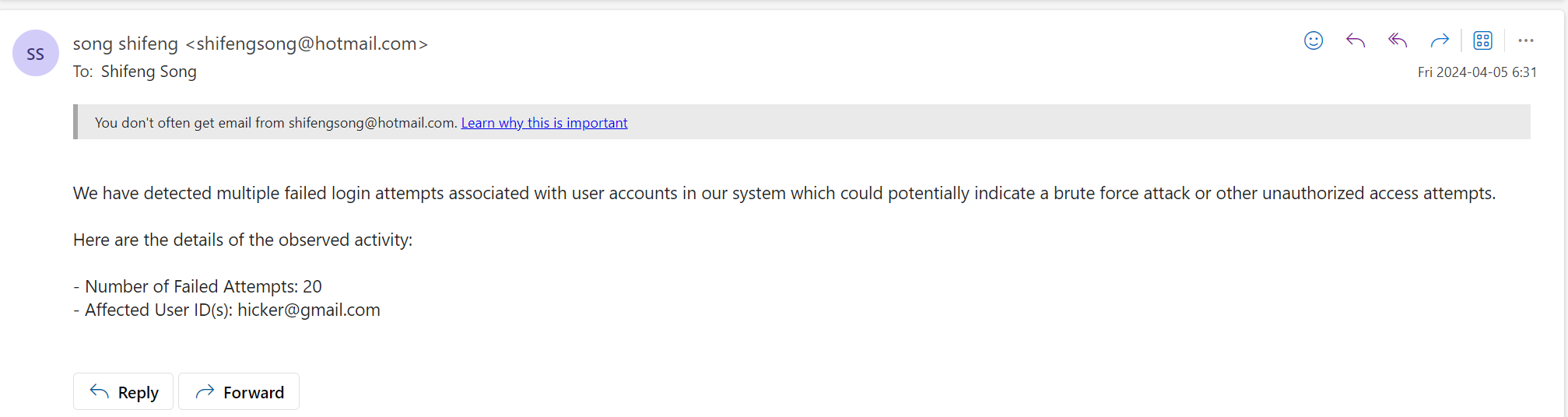
Moreover, this upgraded alert system serves as a deterrent to malicious actors. Knowing that their attempts are likely to trigger an immediate response might discourage persistent unauthorized access attempts. Thus, the system contributes to a safer and more secure digital environment, where users can conduct their activities with confidence, assured that measures are in place to protect their sensitive information from unauthorized access.





If more than five failed login attempts are detected within a two-minute window, the system automatically initiates an email notification process. This alert is sent to designated recipients, informing them of the potential security concern, thus enabling rapid response to mitigate any possible threat.

As the image below showcases the structure of a Logic App designed to automate response actions based on login activity. The process begins with a recurrence trigger, set to activate at specified intervals. Following the trigger, the Logic App executes a pre-defined query that retrieves login attempt results. Once the results are obtained, a 'For each' loop iterates over each entry. If a condition within the loop is met — likely based on login failures — the Logic App proceeds to send an automated email. This email action notifies the concerned parties about potential security issues, allowing for swift corrective measures. The design underscores a proactive approach to monitoring and managing login anomalies.  


The run history of a Logic App named 'AnomalyDetectionApp0138', indicating a successful operation sequence of triggering, querying results, processing each item, and sending an email.  
Email received.  


An alternative approach to identifying anomalous behavior is to consider the geographic location of login attempts. A significant change in the login region within a short timeframe could indicate unauthorized access, as it's unlikely a user would change locations so rapidly under normal circumstances.

However, the implementation of this strategy poses certain challenges. Firstly, acquiring accurate geographic information can be complex, as it often involves analyzing IP addresses and correlating them to physical locations, which requires access to specialized databases and services. Furthermore, such an approach would necessitate changes to the current database schema to include location data for each login attempt.

Due to these complexities and the time constraints faced, it was decided not to include this layer of anomaly detection in the current iteration of the system. Instead, focus was maintained on developing and refining the primary detection mechanisms based on login attempt frequency and failure rates.

Moving forward, the incorporation of geographic analysis into the anomaly detection strategy remains a priority. Future updates aim to enhance the system's robustness by addressing these additional security concerns. The integration of location-based analysis will require careful planning, additional resources, and potentially more advanced technology to ensure accurate and efficient detection of any security breaches.

## Regular Updates:

To maintain a robust cybersecurity posture in the face of evolving threats, it's essential to regularly update detection rules, leveraging the power of machine learning (ML) and artificial intelligence (AI). Begin with comprehensive data collection, integrating diverse sources such as network logs, threat intelligence feeds, and past incident data, ensuring a rich dataset for ML algorithms. This data should be analyzed to extract actionable insights and to identify emerging threat patterns.

Incorporate real-time threat intelligence to stay abreast of the latest attack vectors. Use this intelligence along with insights from past incidents to train and refine your ML models. The objective is to enable these models to learn from new threats and adapt detection mechanisms accordingly. Creating and adjusting detection rules should be an automated, AI-driven process that responds dynamically to the changing threat landscape.

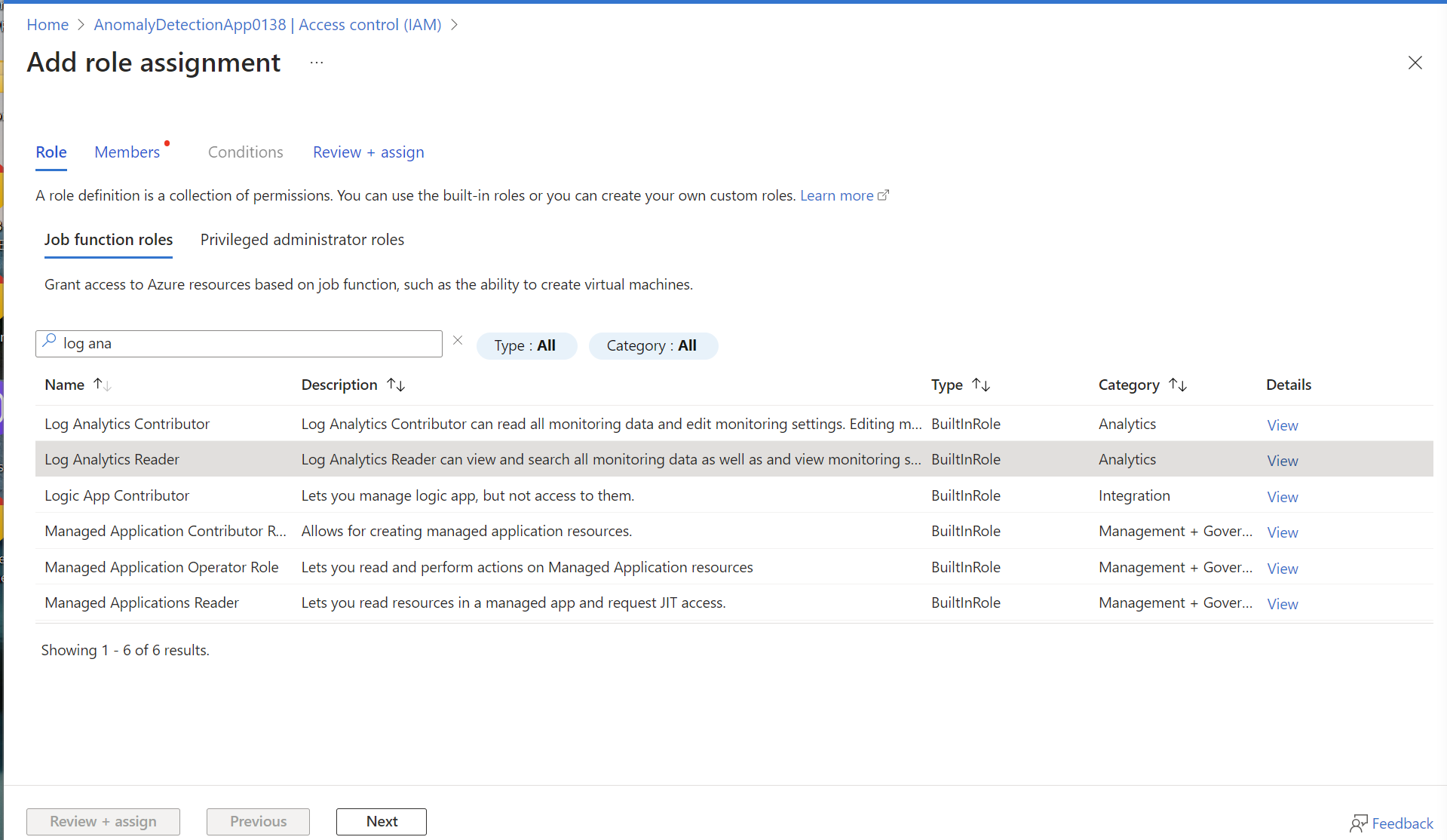
It’s crucial to test new rules and models in a controlled environment to validate their effectiveness, minimizing the risk of false positives that could burden security teams. Once tested, these rules can be deployed to the production environment with a clear process for quick rollback if issues arise.

Continuous monitoring and regular review of the detection rules are imperative. Feedback loops where the system learns from the outcomes of its detections help in fine-tuning accuracy. This learning should feed directly into the incident response process, ensuring that the security team is equipped with the latest information on potential threats.

Documentation and compliance are also key components of this procedure. Every update to the detection rules should be logged, and changes should be communicated to relevant stakeholders, including compliance officers. Finally, ongoing education and training for the security team are vital to ensure they remain knowledgeable about the AI/ML tools at their disposal and the current threat environment.

This iterative and collaborative approach to updating detection rules not only mitigates risk but also enhances the overall resilience of the cybersecurity infrastructure, ensuring preparedness against both current and future threats.

## Least Privilege Access:

Assign the managed identity only the roles that are necessary for the Logic App to perform its tasks. For example, if Logic App only needs to read from log analytics workspace t, assign it the "Log Analytics Reader" role which allows it to read logs and metrics but not to make any changes.  


* Navigate to the logic app in the Azure portal.
* Go to "Access control (IAM)".
* Click on "Add role assignment".
* Select the "Log Analytics Reader" role.
* Assign this role to the managed identity of Logic App.

## Best Practices Explanation:

The solution presented embodies cloud security best practices by focusing on real-time monitoring, anomaly detection, and automated incident response – all of which are core components of a robust cloud security strategy.

Real-time Monitoring and Anomaly Detection

The Logic App's recurrent trigger, set to run at specified intervals, exemplifies proactive monitoring, a cornerstone of cloud security best practices. By consistently analyzing login attempts, the system ensures continuous surveillance of potential security events.

The anomaly detection mechanism, which flags more than five failed login attempts within a two-minute window, adheres to the best practice of defining and implementing security baselines. This threshold is calibrated to balance the need for security with user convenience, acknowledging that occasional failed logins are normal but that an excess may signify a brute force attack or credential stuffing attempt.

Automated Incident Response

When anomalies are detected, the Logic App automatically triggers an email notification. Automated incident response is a recognized best practice, as it ensures that potential security incidents are communicated instantly to relevant stakeholders. It supports a rapid response, which is crucial for minimizing the impact of any security breach.

Principle of Least Privilege

The solution respects the principle of least privilege, with the Logic App granted only the necessary permissions to read from the Log Analytics workspace. This minimizes the risk of privilege escalation and limits the potential damage in the event of a compromised Logic App.

## Incident Response Plan:

Upon the triggering of an alert, the outlined incident response plan is activated, beginning with the immediate notification of the security team via an automated email that provides detailed information on the suspicious login activity. An initial assessment by a security analyst follows to determine the validity and severity of the incident, ensuring it is not a false .

Simultaneously, a comprehensive investigation is launched to trace the source of the failed logins and to understand the breadth of the incident, examining system logs and other forensic data. Remediation efforts are then employed to address the underlying vulnerabilities that led to the breach, such as enforcing password changes, applying necessary security patches, or updating access control policies. Recovery processes are implemented to safely restore normal operations, with vigilant monitoring for any further anomalies. Following the resolution of the incident, the IR team engages in a post-incident review to document the event, analyze the effectiveness of the response, and integrate the insights gained into the organization's ongoing security strategy. This methodical approach is crucial not only for addressing the immediate incident but also for strengthening future defenses and the overall security framework of the organization.

# 

# Source code

The deployment steps for all Logic Apps have been meticulously documented, without the inclusion of any related code snippets.

In this iteration, building on the foundation set by assignment 1, novel approaches were adopted. Unlike the previous assignment, which utilized a syslog server and the Azure log agent, assignment 2 innovatively employed Azure ingestion APIs and Azure Data Collector Endpoints (DCE), as well as Azure Data Collection Rules (DCR), to facilitate the transmission of logs to the Azure Log Analytics workspace. This marked a significant departure from earlier methods and showcased a commitment to exploration and adaptation of new technologies.

However, the code developed for this assignment is not without its imperfections. For instance, sensitive information, such as keys, are hardcoded within the codebase. The best practice, and a critical area for improvement, would be to externalize these credentials. This can be achieved through the utilization of Azure Key Vault for securely storing these sensitive elements. Azure Key Vault provides a centralized and secure store that can be leveraged to dynamically manage secrets, such as API keys and other credentials, at runtime. By referencing these secrets in the code via Key Vault, rather than including them explicitly, the security of the application is significantly enhanced. This practice not only minimizes the risk of credential leakage but also simplifies the process of key rotation and management. Implementing this change would be a key step forward in aligning with cloud security best practices, ensuring that the application remains robust, secure, and easier to maintain..

By securing sensitive data through Azure Key Vault, a managed service designed to safeguard cryptographic keys and other secrets used by cloud apps and services, a more secure and scalable solution can be realized. The Key Vault integration not only helps in protecting against data exposure but also facilitates a more streamlined approach to secret management, which is pivotal in a cloud-centric infrastructure.

# References

<https://www.crowdstrike.com/guides/azure-logging/log-ingestion/>  
  
<https://jasonwatmore.com/post/2020/02/01/react-fetch-http-post-request-examples>