## Lab Report: CST8912 – Cloud Solution Architecture, Graded Lab Activity #10

**Submitted by:** ZheZhang041109657
**Submitted to:** Prof. Tanishq Bansal
**Date:** Mar 27, 2025

---

**Title**

Automating Alerts and Monitoring in Azure Logic Apps for SQL Database and Blob Storage

---

**Introduction or Purpose**

The purpose of this lab was to explore and implement cloud-based automation workflows using Azure Logic Apps, a Platform as a Service (PaaS) offering from Microsoft Azure. The main objectives were to create automated workflows for alerting users based on data updates in a SQL Database, monitoring file uploads in Azure Blob Storage, and analyzing workflow performance using Azure Monitor. Specifically, the lab involved:

1. Creating a Logic App to send email notifications when specific data is present in a SQL Database.
2. Designing a Logic App to alert users when a file is not uploaded to a specific folder in Blob Storage by a certain time.
3. Monitoring the workflows using Azure Monitor.
4. Cleaning up all resources created during the lab.

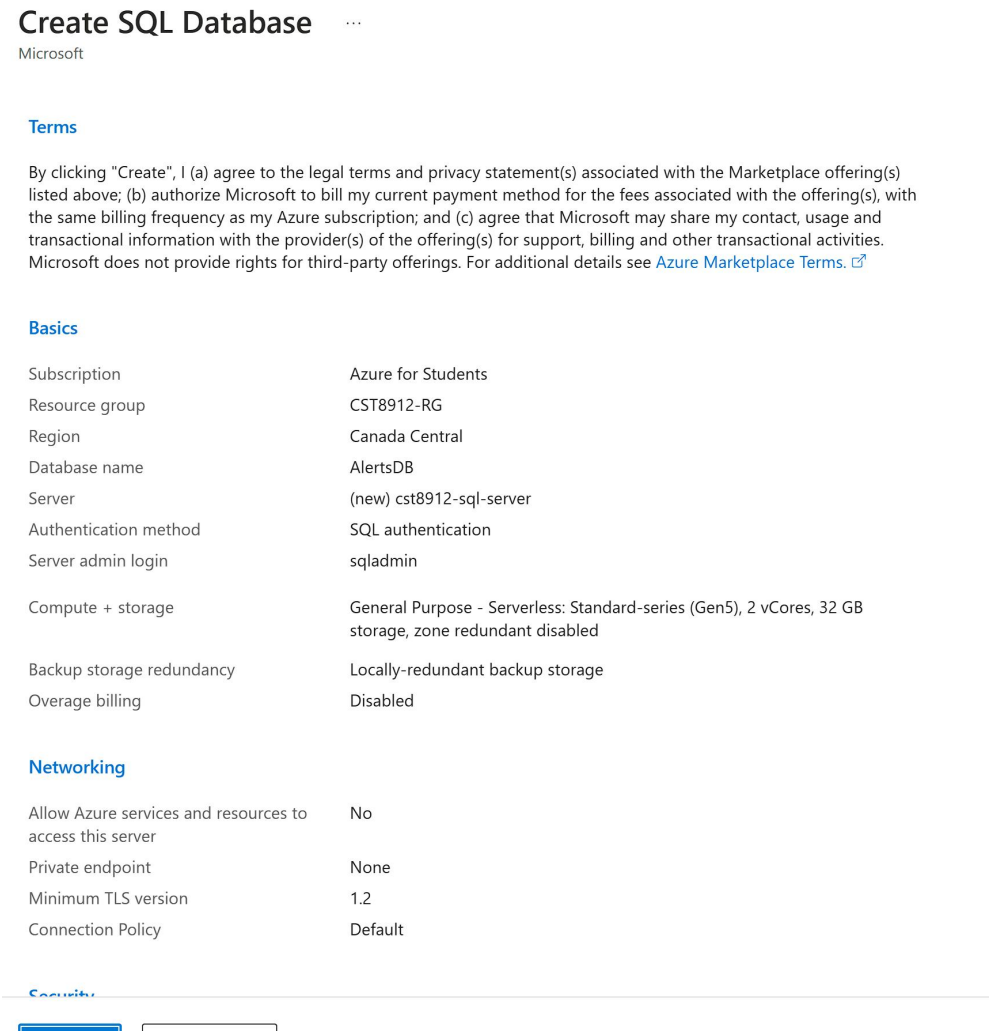
---

**Steps Covered in the Lab**

**Task 1: Alert Users Based on Data in SQL Database**

1. **Created Logic App and SQL Database**:
    1. Created a Logic App named AlertsLogicApp with a consumption-based plan in the Canada Central region.

2.Created a SQL Database instance named AlertsDB in the Canada Central region, with a server named cst8912-sql-server and admin credentials (sqladmin, password: P@ssw0rd123!).



Create SQL Database ...

Microsoft

**Terms**

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For additional details see Azure Marketplace Terms.

**Basics**

| | |
|---|---|
| Subscription | Azure for Students |
| Resource group | CST8912-RG |
| Region | Canada Central |
| Database name | AlertsDB |
| Server | (new) cst8912-sql-server |
| Authentication method | SQL authentication |
| Server admin login | sqladmin |
| Compute + storage | General Purpose - Serverless: Standard-series (Gen5), 2 vCores, 32 GB storage, zone redundant disabled |
| Backup storage redundancy | Locally-redundant backup storage |
| Overage billing | Disabled |

**Networking**

| | |
|---|---|
| Allow Azure services and resources to access this server | No |
| Private endpoint | None |
| Minimum TLS version | 1.2 |
| Connection Policy | Default |

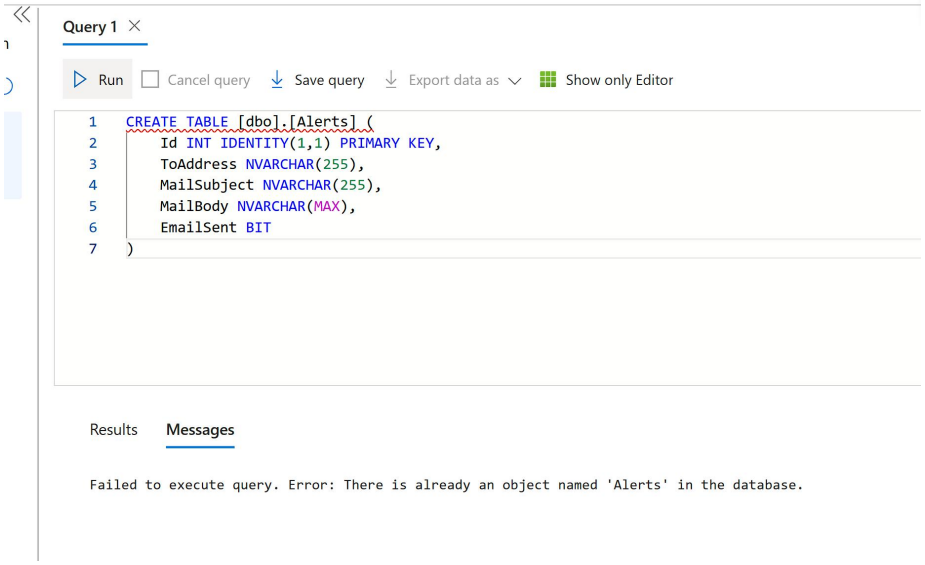
Security

2. **Created Alerts Table**:

1.Used the Query Editor in Azure SQL to create a table named [dbo].[Alerts] with columns: Id, ToAddress, MailSubject, MailBody, and EmailSent.

2.SQL Command:

```
CREATE TABLE [dbo].[Alerts] ( Id INT IDENTITY(1,1) PRIMARY KEY, ToAddress NVARCHAR(255), MailSubject NVARCHAR(255), MailBody NVARCHAR(MAX), EmailSent BIT )
```



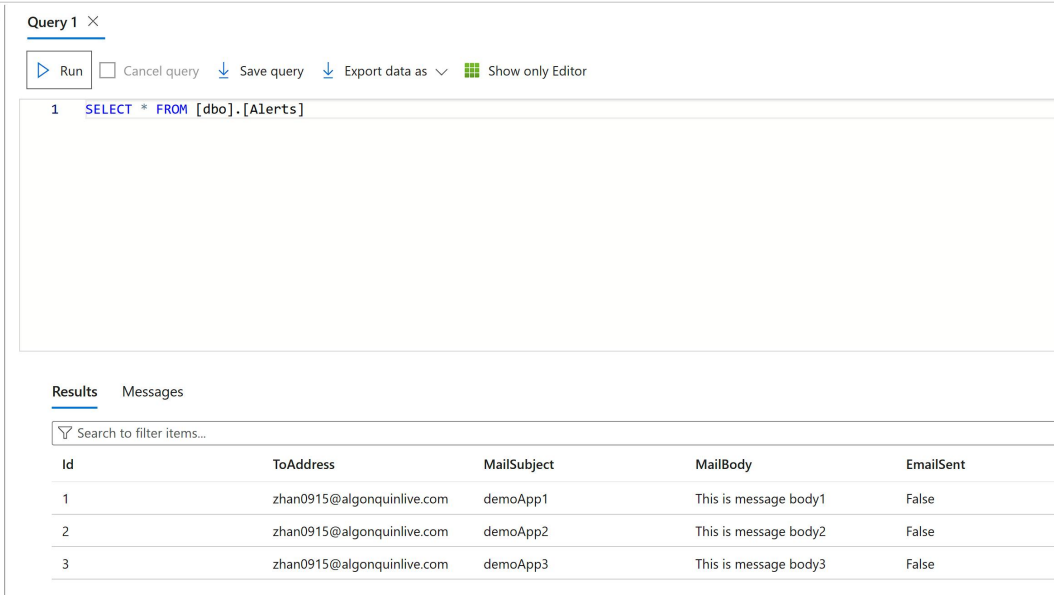

.

3. **Inserted Records**:

1.Inserted three records into the [dbo].[Alerts] table:

sql

```
INSERT INTO [dbo].[Alerts] (ToAddress, MailSubject, MailBody, EmailSent)
VALUES ('zhan0915@algonquinlive.com', 'demoApp1', 'This is message body1',
0); INSERT INTO [dbo].[Alerts] (ToAddress, MailSubject, MailBody, EmailSent
VALUES ('zhan0915@algonquinlive.com', 'demoApp2', 'This is message body2',
0); INSERT INTO [dbo].[Alerts] (ToAddress, MailSubject, MailBody, EmailSent
VALUES ('zhan0915@algonquinlive.com', 'demoApp3', 'This is message body3',
0);
```




4. **Configured Logic App**:

    1. Added a **"Recurrence"** trigger with an interval of 3 minutes.

    2. Added a **"Get rows (V2)"** action to retrieve rows from [dbo].[Alerts] in AlertsDB.

    3. Added a **"For each"** loop to iterate over the rows (using the value output from **"Get rows (V2)"**).

    4. Added a **"Send an email (V2)"** action to send emails using Office 365 Outlook, with:

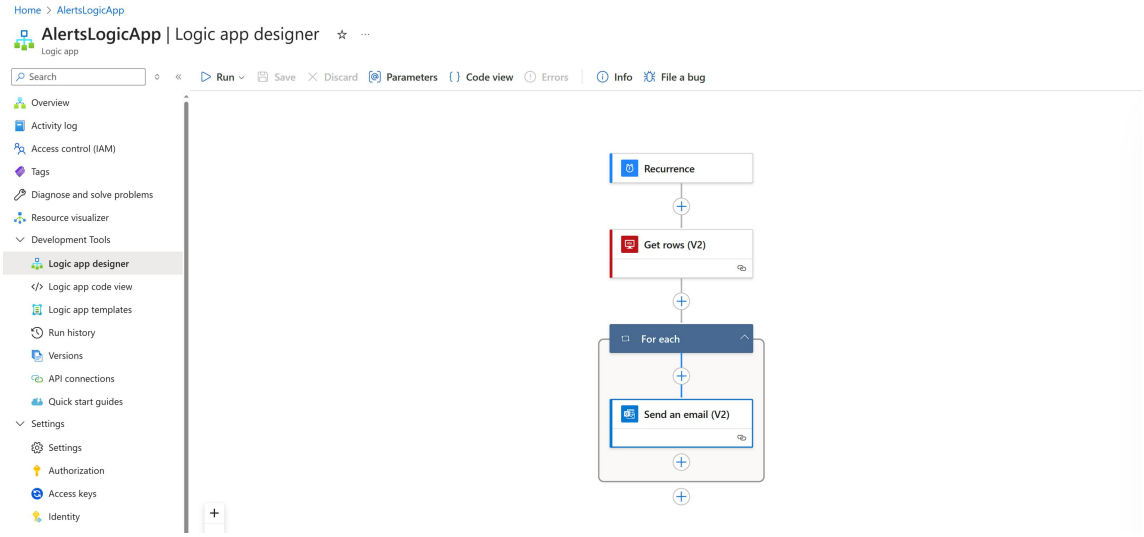
        1. **To**: ToAddress (dynamic content).

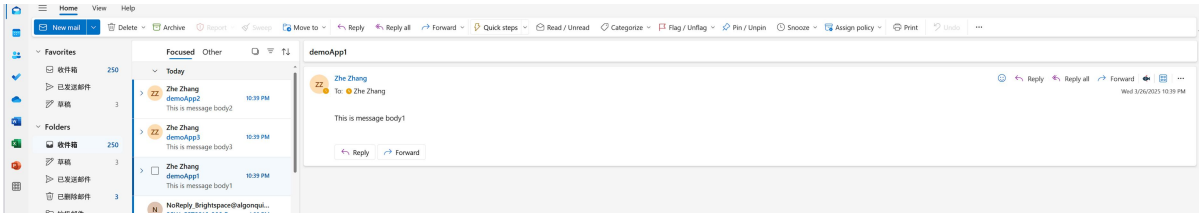
        2. **Subject**: MailSubject (dynamic content).

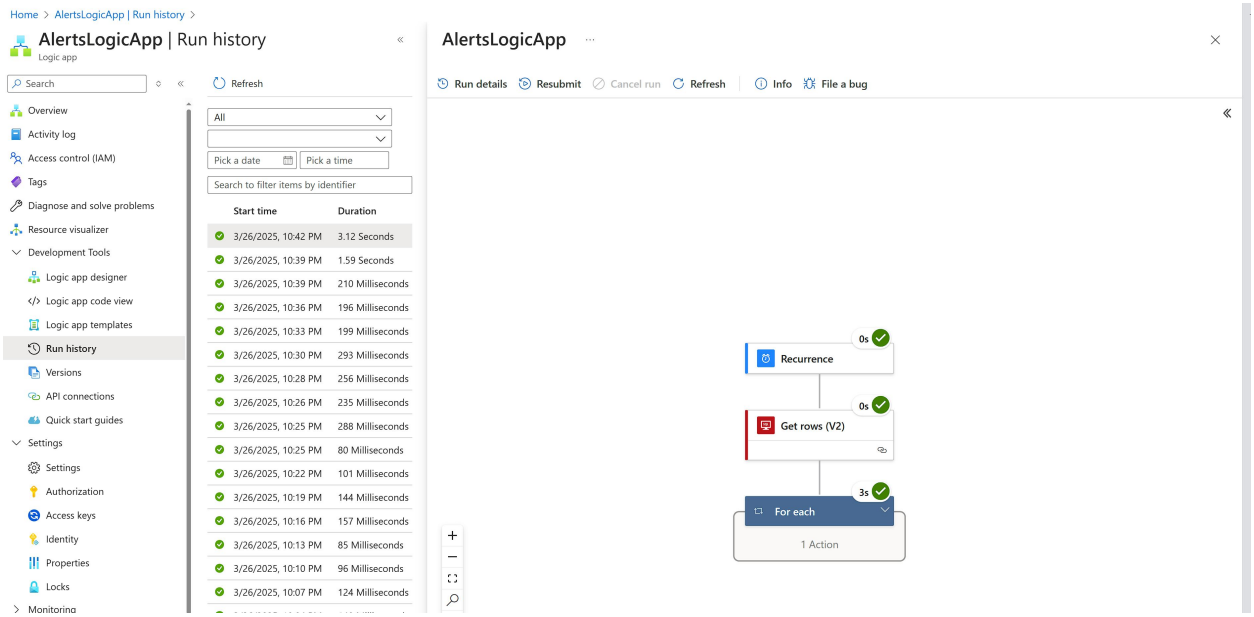
        3. **Body**: MailBody (dynamic content).

5. **Tested Logic App**:

    1. Ran the Logic App and confirmed that three emails were sent to zhan0915@algonquinlive.com with the expected subjects and bodies.



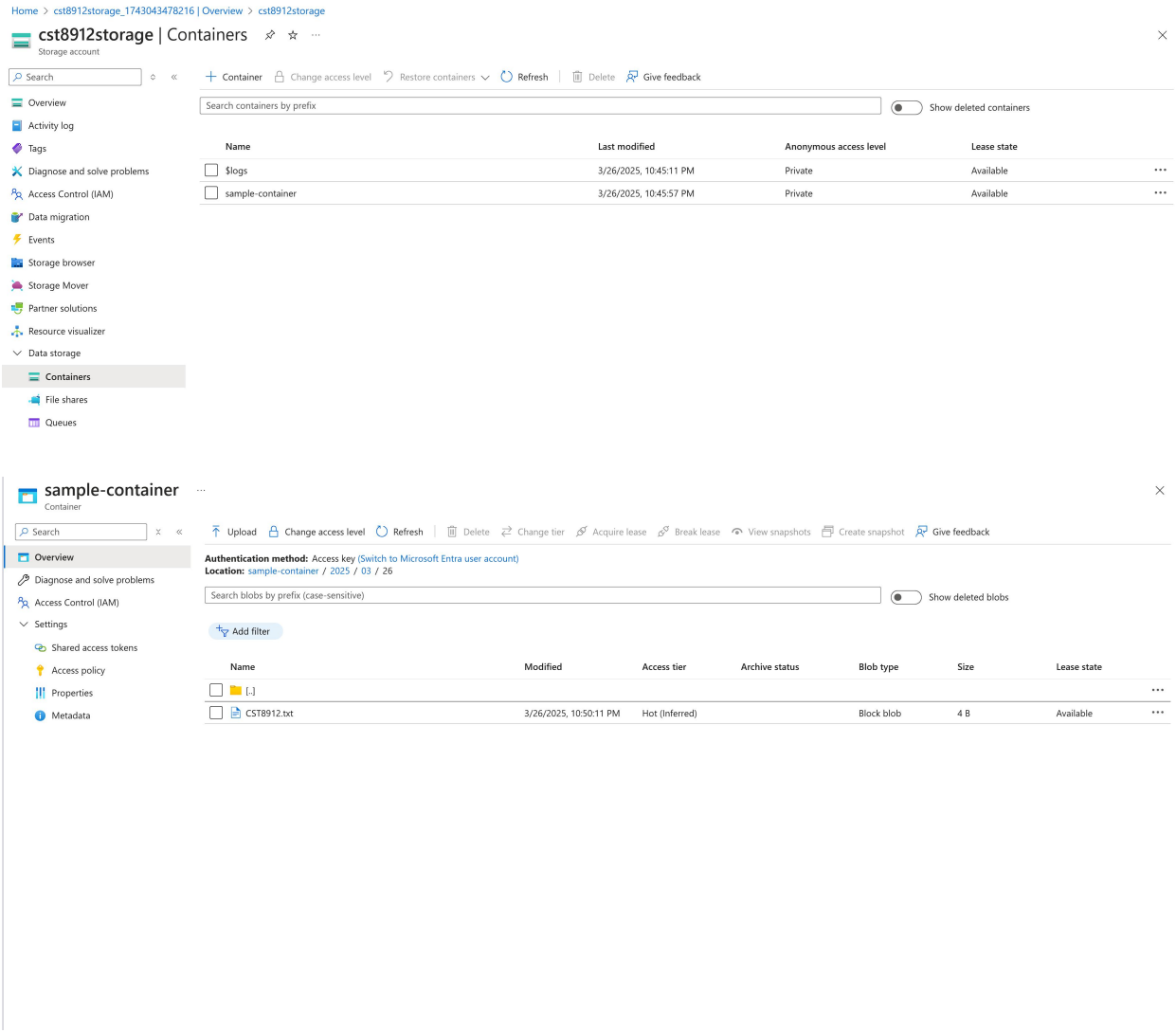

## Task 2: Alert Users When File Not Uploaded to Blob Storage

1. **Created Storage Account:**

1.Created a storage account named cst8912storage in the Canada Central region.

2.Created a container named sample-container with private access.

3.Created a folder structure /2025/03/26 and uploaded a test file (test.txt).




2. **Created Logic App:**

1.Created a new Logic App named FileCheckLogicApp in the Canada Central region.

2.Added a **"Recurrence"** trigger to run daily at 23:00 Eastern Time (UTC 04:00 next day).

3.Added a **"Lists blobs (V2)"** action to check the folder /2025/03/26 (temporarily set as a static value for testing).

4.Added a **"Condition"** action to check if the blob list is empty:

1. Condition: length(body('Lists_blobs_(V2)')?['value']) is equal to 0.

5. In the **"If true"** branch, added a **"Send an email (V2)"** action:

    1. **To**: zhan0915@algonquinlive.com.

    2. **Subject**: File Not Uploaded.

    3. **Body**: concat('No file uploaded to /', utcNow('yyyy/MM/dd'), ' folder today.').

6. **"If false"** branch was left empty.

3. **Tested Logic App:**

    1. Encountered 404 ContainerNotFound errors due to issues with the container. Recreated sample-container and reconfigured the connection.

    2. Tested with /2025/03/26 folder empty: Received the "File Not Uploaded" email.

    3. Tested with /2025/03/26 folder containing a file: No email was sent, as expected.

**Task 3: Monitor Workflows in Azure Monitor**

1. **Enabled Diagnostic Logs:**

    1. In AlertsLogicApp, enabled diagnostic settings to send **"WorkflowRuntime"** logs to a Log Analytics workspace (LogicAppWorkspace).

2. **Ran Query in Azure Monitor:**

    1. Navigated to Azure Monitor > **"Logs"**.

    2. Selected AlertsLogicApp as the scope.

    3. Ran the query:

    kql

    ```
    AzureDiagnostics | where Category == "WorkflowRuntime"
    ```

    3. Initially, no data was found because logs were not configured. After enabling diagnostics and running the Logic App, logs were available.

**Task 4: Clean Resources**

1. **Deleted Resource Group：**

    1.Navigated to **"Resource groups"**, selected CST8912-RG.

    2.Clicked **"Delete resource group"**, confirmed by entering CST8912-RG, and clicked **"Delete"**.

---

**Results**

The lab successfully demonstrated the use of Azure Logic Apps to automate workflows for SQL Database and Blob Storage monitoring. Key findings include:

- Task 1: The Logic App successfully sent email notifications based on SQL Database data, though initial connection issues with SQL Server required firewall adjustments.
- Task 2: The Logic App correctly detected missing files in Blob Storage and sent notifications, but container configuration issues caused 404 ContainerNotFound errors, which were resolved by recreating the container.
- Task 3: Azure Monitor provided insights into workflow performance after enabling diagnostic logs.
- Task 4: All resources were cleaned up by deleting the resource group.

---

**References**


- Microsoft Azure Documentation: Azure Logic Apps Overview
- Microsoft Azure Documentation: Azure Monitor Logs