

Azure Data Lake Storage – Configuring Notification Alerts

Prepared by

Data SQL Ninja Engineering Team (datasqlninja@microsoft.com)

Disclaimer

The High-Level Architecture, Migration Dispositions and guidelines in this document is developed in consultation and collaboration with Microsoft Corporation technical architects. Because Microsoft must respond to changing market conditions, this document should not be interpreted as an invitation to contract or a commitment on the part of Microsoft.

Microsoft has provided generic high-level guidance in this document with the understanding that MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION CONTAINED HEREIN.

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2019 Microsoft. All rights reserved.

Note: The detail provided in this document has been harvested as part of a customer engagement sponsored through the [Data SQL Ninja Engineering](#).

Table of Contents

1	Introduction	5
2	Pre-requisites	6
3	ADLS Event Notification Alerts using Log Analytics.....	7
3.1	Configure/Enable Diagnostics	8
3.2	Define Action Group	9
3.3	Using Azure Monitor configure alerts from Log Analytics	10
3.4	Verify the Alerts	14
3.5	Log Analytics Approach – Things to remember!	15
4	ADLS Event Notification Alerts using Event Hub.....	16
4.1	Configure Event Hub namespace	17
4.2	Configure ADLS Diagnostics.....	20
4.3	Define Azure Function to filter ADLS events.....	22
5	ADLS Event Notification Alerts using Event Grid.....	27
5	Appendices.....	28
5.1	Appendix - External References	28
6	Feedback and suggestions	29

Table of Figures

Figure 1: Log Analytics Approach	7
Figure 2: Turn-on diagnostics.....	8
Figure 3: Turn-on diagnostics - Choose a Storage Account	8
Figure 4: ADLS Gen1 Diagnostics Settings Overview	9
Figure 5: Add action group.....	9
Figure 6: Add Action Group detail.....	10
Figure 7: Azure Monitor - Alerts View	10
Figure 8: Custom Log Search Condition.....	11
Figure 9: Log Analytics - Alert Logic Configuration.....	11
Figure 10: Action Group Assignment	12
Figure 11: Action Group Selection	12
Figure 12: Create Rule/Alert Details.....	13
Figure 13: Manage Rules/Verify New Rule.....	13
Figure 14: Webhook Alert Sample.....	14
Figure 15: Email Alert Sample.....	15
Figure 16: Event Notification Alerts using Event Hub	16
Figure 17: Event Hubs	17
Figure 18: Create Event Hub Namespace	18
Figure 19: Resources Overview - Event Hub Namespace Listing	19
Figure 20: Create Event Hub	19
Figure 21: Selecting Event Hub Namespace	20
Figure 22: Diagnostic Settings Overview.....	21
Figure 23: Create Function App	22
Figure 24: Existing Function Apps Overview.....	23
Figure 25: Define Azure Function (In-portal template)	23
Figure 26: Azure Function - More Templates.....	24
Figure 27: Azure Functions - Azure Event Hub Trigger	24
Figure 28: MicrosoftAzure.WebJobs.Extensions.EventHubs Extension	25
Figure 29: Sample Code generated by Designer	26
Figure 30: Generated Code customization sample	26
Figure 31: Event Notification Alerts using Event Grid	27

1 Introduction

Azure Data Lake Storage (ADLS) is a secure, durable, cost-effective and highly scalable cloud data lake offering from Microsoft. Besides providing file based I/O operations, it offers a deep integration with Azure PaaS and SaaS offerings such as Azure Data Factory, Azure Databricks, Azure HDInsight, Azure SQL Datawarehouse and PowerBI. To ensure security, ADLS is well integrated with Azure Active Directory service that allow customers to configure ACL-based roles that match their functional requirements.

This document is primarily focused on one of the key operational requirements – be able to receive notifications and alerts whenever things change on ADLS, using one of the following solution approaches:

- Event Notification & Metrics from ADLS using Log Analytics
- Event Notification & Metrics from ADLS using Event Hub
- Event Notification & Metrics from ADLS using Event Grid

2 Pre-requisites

It is important that you have following available or created before proceeding to further sections below:

- Resource Group, for example - ``adlsalerts_rg`` (``rg`` → Resource Group)
- ADLS Gen 1 Resource, for example - ``adlsalertsgen1`` (``..gen1`` → ADLS Gen1)
- Storage Account Resource, for example - ``adlsgen1alertssa`` (``..sa`` → Storage Account)
- Log Analytics Resource, for example - ``adlsgen1alertsLAWS`` (``..LAWS`` → Log Analytics Workspace)
- Monitor → Alerts → Action Group, for example - ``ADLSGen1AlertsActionGroup`` (see section further below in this documentation on how to go about creating an action group)

Please review Azure Portal documentation [here](#), on how to approach such definitions, except for action group. There's a brief section further below that describes necessary steps with references to additional detail.

3 ADLS Event Notification Alerts using Log Analytics

This section describes the steps involved in gathering diagnostics logs from Azure Data lake account into Azure log analytics workspace and generating webhook and email alert thereafter.

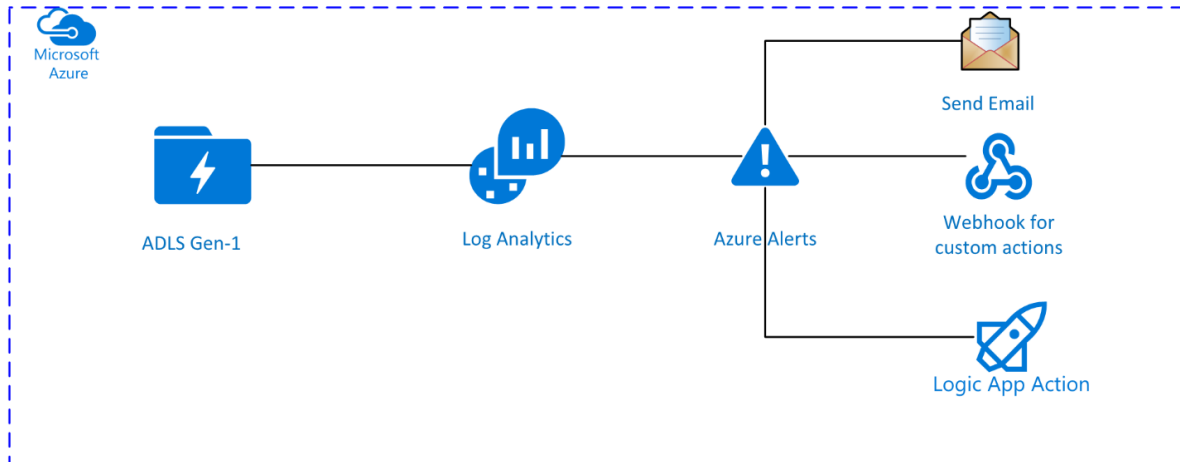


Figure 1: Log Analytics Approach

Following is the high-level outline of steps involved using Log Analytics approach:

- Configure/Enable Diagnostics to point to Azure Log Analytics
- Using Azure Monitor configure alerts from Log Analytics
- Pros and Cons of the approach

3.1 Configure/Enable Diagnostics

As a pre-requisite one must enable or turn-on the diagnostics for the corresponding ADLS Account. Use the Azure Portal Interface to achieve this, as shown in the following visual:

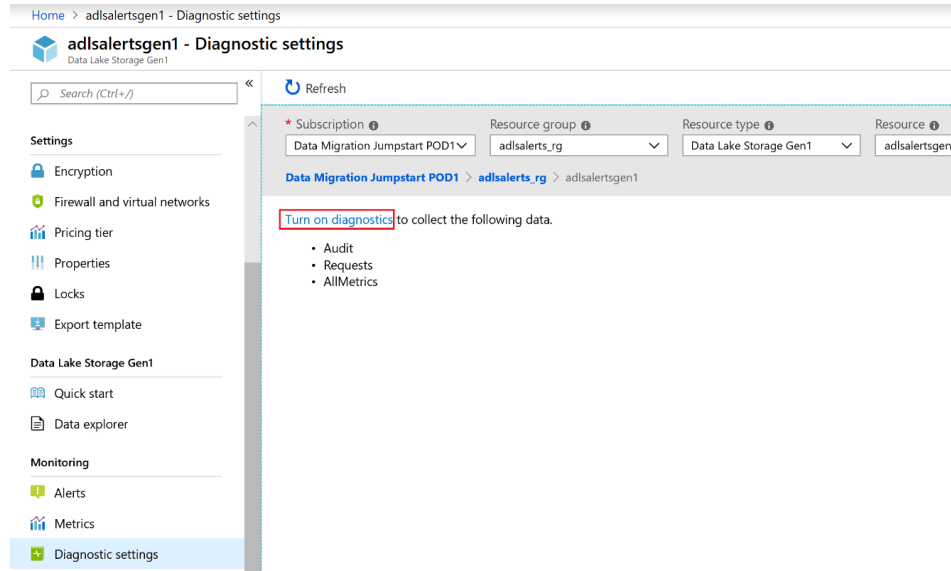


Figure 2: Turn-on diagnostics

Click on Turn-on diagnostics. In the subsequent screen as shown below, select storage account and log analytics target. Storage account is used to archive the diagnostic events like create, delete, etc. Log Analytics will collect these events as submitted by ADLS setting and will be available for further triage and reasoning.

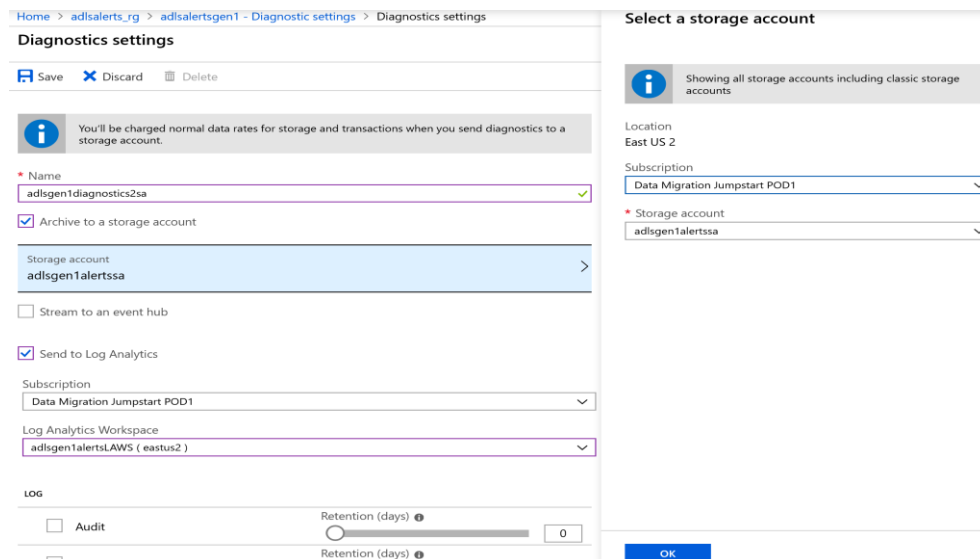


Figure 3: Turn-on diagnostics - Choose a Storage Account

Hit save once you have made necessary configuration changes. Following screen will now list the diagnostic configuration that you just created:

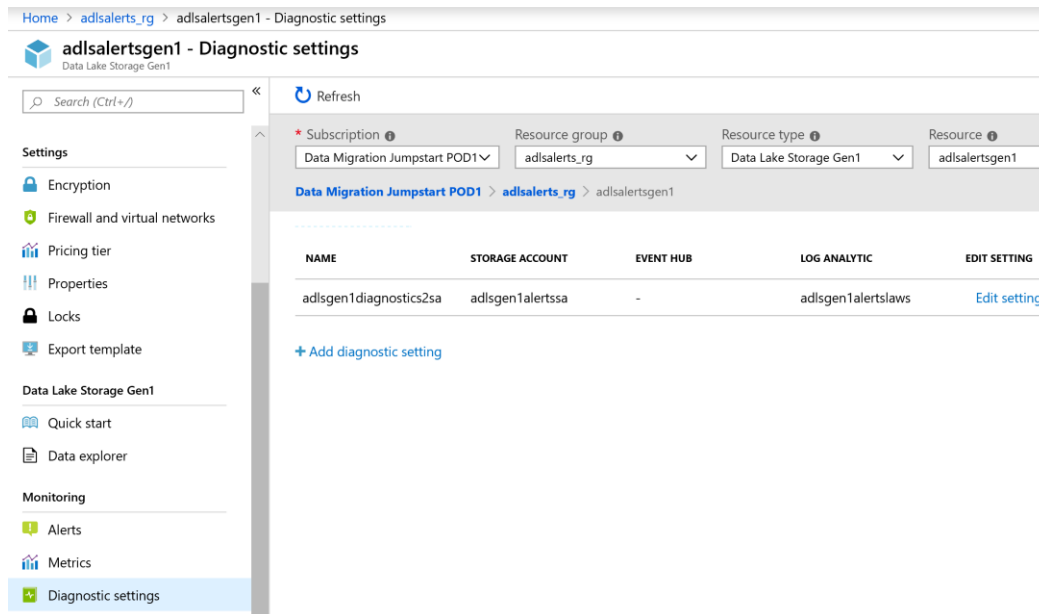


Figure 4: ADLS Gen1 Diagnostics Settings Overview

3.2 Define Action Group

Next step is to configure Action Group! Action groups abstract the alert outcomes as actions. [Here](#) is a quick link to Azure Portal documentation that describes how to manage Action Groups.

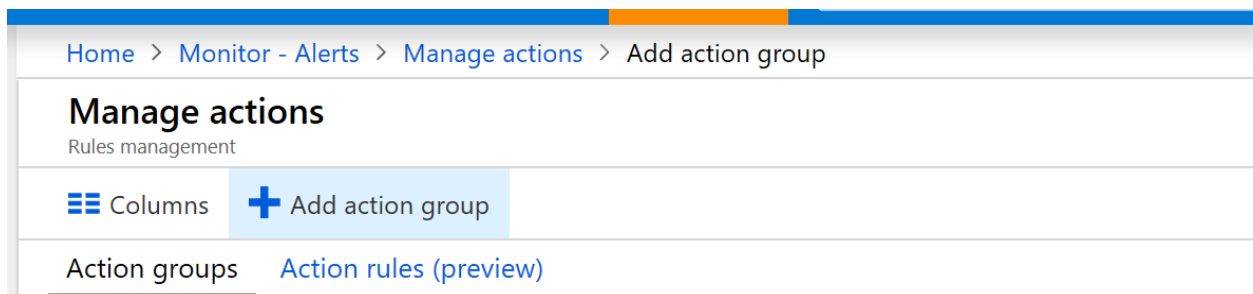


Figure 5: Add action group

Following visual provides the summary detail of actions that this new action group would abstract – an email and a web hook.

Figure 6: Add Action Group detail

For a test webhook, you can leverage defining a webhook end point using the publicly available developer interface – www.webhook.site URL. A sample URL will look like this - <http://webhook.site/45735651-caa0-46a9-9ea1-64109ddeafe2>.

3.3 Using Azure Monitor configure alerts from Log Analytics

In the previous step when you have saved the diagnostics configuration, wait for a while, say about few minutes. This will allow the backend to enable a key schema in the log analytics called `AzureDiagnostics`. In this section we will go through how to query that schema for events of our interest using Log Analytics query interface. Following visual shows how to navigate to the Alerts configuration feature (path to follow is Home → Monitor → Alerts tab):

Figure 7: Azure Monitor - Alerts View

Click on new alert rule option as highlighted above. In the subsequent screen we will define a condition using a custom log search option, as show in the following visual:

Home > Monitor - Alerts > Create rule

Configure signal logic

Choose a signal below and configure the logic on the next screen to define the alert condition.

All signals (153)

Signal type: All Monitor service: All

Search by signal name

SIGNAL NAME	SIGNAL TYPE	MONITOR SERVICE
Custom log search	Log	Log analytics
All Events	Log(Saved Query)	Log analytics
Count of Events containing the word "started" grouped by EventID	Log(Saved Query)	Log analytics
Count of Events grouped by Event Log	Log(Saved Query)	Log analytics
Count of Events grouped by Event Source	Log(Saved Query)	Log analytics

Figure 8: Custom Log Search Condition

Let us now look at the details of this search criteria:

Create rule

Rules management

*** RESOURCE**

adlsgen1alertsLAWS

Select

*** CONDITION**

No condition defined, click Add

ACTIONS

No actions defined. Add

ALERT DETAILS

* Alert rule name: Specify alert rule name. Save

Description: Specify alert description here

Enable rule upon creation: Yes No

Create alert rule

Configure signal logic

Custom log search

100
80
60
40
20
0

10:45 AM 10:50 AM 10:55 AM 11 AM 11:05 AM

* Search query

AzureDiagnostics
| where OperationName in ("mkdirs", "append", "delete")

Query to be executed : AzureDiagnostics | where OperationName in ("mkdirs", "append", "delete") | count
For time window : 5/5/2019, 11:01:41 AM - 5/5/2019, 11:06:41 AM

Alert logic

Based on: Number of results Operator: Greater than Threshold value: 0

Condition preview

Whenever the custom log search is greater than 0 count

Evaluated based on

* Period (in minutes): 5 * Frequency (in minutes): 5

Done

Figure 9: Log Analytics - Alert Logic Configuration

It is important to understand, the minimum latency to receive and process alerts is 5 minutes, for up to 24 hours.

Create rule

Rules management

RESOURCE

adlsgen1alertsLAWS

Select

HIERARCHY

Data Migration Jumpstart POD1 > adlsalerts_rg

CONDITION

✓

Whenever the Custom log search is Greater than 0 count

Monthly cost in USD (Estimated)

\$ 1.50

Add

Total \$ 1.50

We currently support configuring only two metrics signals or one log search signal or one activity log signal per alert rule. An alert will be triggered when the conditions for all the above configured criteria are met

ACTIONS

No actions defined.

Add

Customize Actions

☐ Email subject

☐ Include custom Json payload for webhook

Figure 10: Action Group Assignment

When you click on Add above, it will open an interface to select the sample Action Group that we have defined in prior sections. Here's the visual:

Select an action group to attach to this alert rule

Action rules (Preview)

Configure this action across resources in this scope using Action rules (preview). Action rules allows you to set granular control of notifications, suppression and run diagnostics for quick troubleshooting. [Learn more](#)

Create action rule

For metric and log alerts, action groups selected must be in the alert rule's subscription. For activity log alerts, action groups can be selected from subscriptions other than the alert rule's subscription.

Subscription

Data Migration Jumpstart POD1

Search to filter items...

ACTION GROUP NAME

CONTAIN ACTIONS

✓

ADLSGen1AlertsActionGroup

1 Email, 1 Webhook

mygroup

1 Email

BlobAlert

1 Email

TestWeebhook

1 Webhook

Figure 11: Action Group Selection

Prepared by Data SQL Ninja Engineering Team

Once you have added the Action Group under the Actions option of the Create Rule interface, the next step is to add Alert details on the same interface. This will allow you complete the last step in the sequence and submit to create the alert rule.

ALERT DETAILS

* Alert rule name ⓘ

ADLS Gen1 File Ops ✓

Description

Tracked option is mkdirs/append/delete. |

* Severity ⓘ

Warning(Sev 1) ▼

Enable rule upon creation

Yes

No

☐ Suppress Alerts ⓘ

Create alert rule

Figure 12: Create Rule/Alert Details

You can verify the newly defined Alert rule, by navigating to the Home → Monitor – Alerts tab and by selecting the tab Manage Rules:

Home > Monitor - Alerts > Rules

Rules

Rules management

+ New alert rule | Edit columns | Manage action groups | View classic alerts | Refresh | Migrate to new rules | Enable | Disable | Delete

* Subscription ⓘ: Data Migration Jumpstart POD1 | Resource group ⓘ: adisalerts_rg | Resource type ⓘ: 4 selected | Resource ⓘ: adisgen1alertsLAWS | Signal type ⓘ: All sources | Status ⓘ: Enabled

Data Migration Jumpstart POD1 > adisalerts_rg > adisgen1alertsLAWS

i Alerts(Classic) will be retired on June 30th. Use the voluntary migration tool to upgrade to the faster, simpler, and more scalable metric alerts platform. [Know more](#)

Displaying 1 - 1 rules out of total 1 rules

Search alert rules based on rule name and condition...

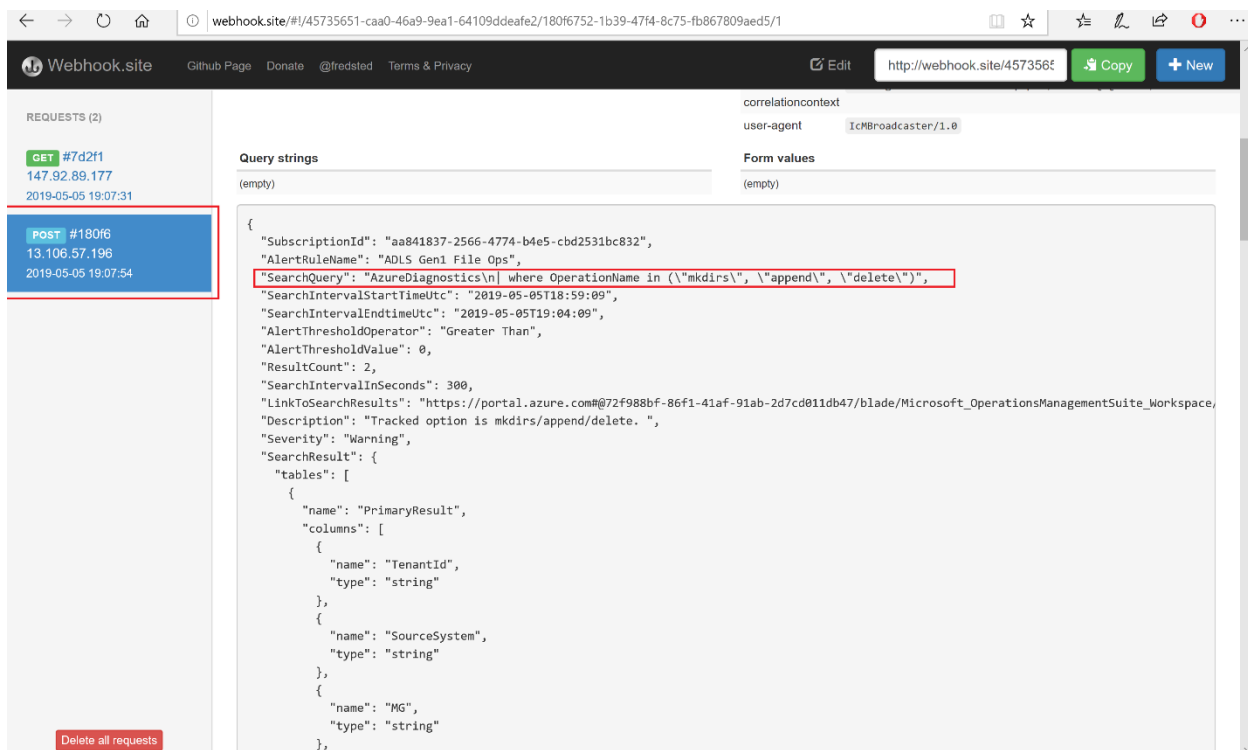
NAME	CONDITION	STATUS	TARGET RESOURCE	TARGET RESOURCE TYPE	SIGNAL TYPE
ADLS Gen1 File Ops	AzureDiagnostics where OperationName in ("mkdirs", "append", "delete")	Enabled	adisgen1alertsLAWS	Log Analytics workspaces	Log Search

Figure 13: Manage Rules/Verify New Rule

3.4 Verify the Alerts

Recommend that you wait about 5 minutes, to start observing the action outcomes. Remember the SLA set above to 5 minutes minimum latency defined with an event duration of since last 5 minutes up to a maximum of past 24 hours. To trigger an alert you can use Azure Data Explorer client and create few folders under the same subscription POD and the ADLS Gen1 resource that is part of the resource group where you have defined the alerts. The ADLS resource must be same as configured to be monitored by the alert's action group.

Here's the sample of the webhook result (received as a POST request):

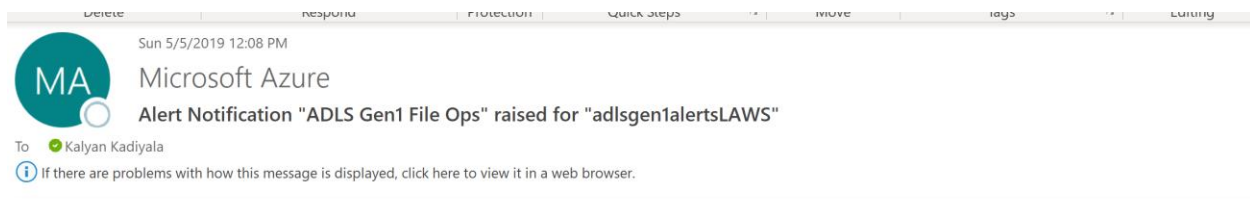


The screenshot shows the Webhook.site interface. On the left, a list of requests is shown, with a POST request #180f6 highlighted. The main area displays the details of this request, including the query strings (empty) and the form values (empty). The JSON body of the request is shown, containing information about the alert rule and search results. A red box highlights the 'SearchQuery' field in the JSON body.

```
{
  "SubscriptionId": "aa841837-2566-4774-b4e5-cbd2531bc832",
  "AlertRuleName": "ADLS Gen1 File Ops",
  "SearchQuery": "AzureDiagnostics\\n| where OperationName in (\\\"mkdirs\\\", \\\"append\\\", \\\"delete\\\")",
  "SearchIntervalStartTimeUtc": "2019-05-05T18:59:09",
  "SearchIntervalEndTimeUtc": "2019-05-05T19:04:09",
  "AlertThresholdOperator": "Greater Than",
  "AlertThresholdValue": 0,
  "ResultCount": 2,
  "SearchIntervalInSeconds": 300,
  "LinkToSearchResults": "https://portal.azure.com#@72f988bf-86f1-41af-91ab-2d7cd011db47/blade/Microsoft_OperationsManagementSuite_Workspace",
  "Description": "Tracked option is mkdirs/append/delete. ",
  "Severity": "Warning",
  "SearchResult": {
    "tables": [
      {
        "name": "PrimaryResult",
        "columns": [
          {
            "name": "TenantId",
            "type": "string"
          },
          {
            "name": "SourceSystem",
            "type": "string"
          },
          {
            "name": "MG",
            "type": "string"
          }
        ]
      }
    ]
  }
}
```

Figure 14: Webhook Alert Sample

Here is the email sample of the same alert:



⚠️ Your Azure Monitor alert was triggered

We are notifying you because there are 2 counts of "ADLS Gen1 File Ops".

Essentials

Name	ADLS Gen1 File Ops
Severity	Warning
Resource	adlsgen1alertsLAWS
Search interval start time	May 5, 2019 18:59:09 UTC
Search interval duration	5 min
Search query	AzureDiagnostics where OperationName in ("mkdirs", "append", "delete")
Search results	2 result(s)

Figure 15: Email Alert Sample

3.5 Log Analytics Approach – Things to remember!

As you observe the approach offers a pure configuration-based approach to define notification alerts for corresponding ADLS operations – mkdir, append and delete. The query interface supports a wide variety of query language features. Here is the [link](#) to learn more! The only caveat you need to assert for your applicability is – is the minimum lag duration of 5 minutes is agreeable for your notification alerts use case? Else, this is a great way to setup and receive Notification alerts using one of the available channels – Email, Webhook or SMS alerts.

4 ADLS Event Notification Alerts using Event Hub

The next solution approach that we will review is the ability to use Event Hub to receive notification alerts for select events generated by activities on your configured ADLS Gen1 resource. Here's the empirical view of the flow:

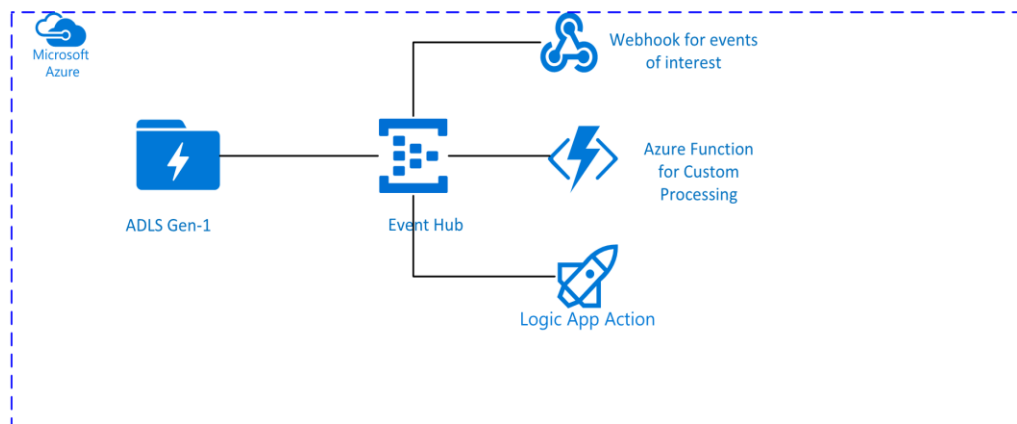


Figure 16: Event Notification Alerts using Event Hub

Briefly, the steps involved in this approach include:

- Configure ADLS Diagnostics to point to Event Hub as target destination to log events.
 - Pre-requisite step here is to define the Event Hub namespace.
- Create Azure Function to ingest and filter ADLS logs.
- (Optional) Create a Logic App with Event Hub as Trigger.

4.1 Configure Event Hub namespace

In this section we will see the approach to configure an Event Hub to capture events from ADLS resource. For the context, let us name our Event Hub namespace as *adlsgen1alerts2ehns*. From your corresponding Resource Group (for example – *adlsalerts_rg*) blade i.e., pane where resources of a Resource Group can be viewed, click on Add a resource icon on the top navigation pane.

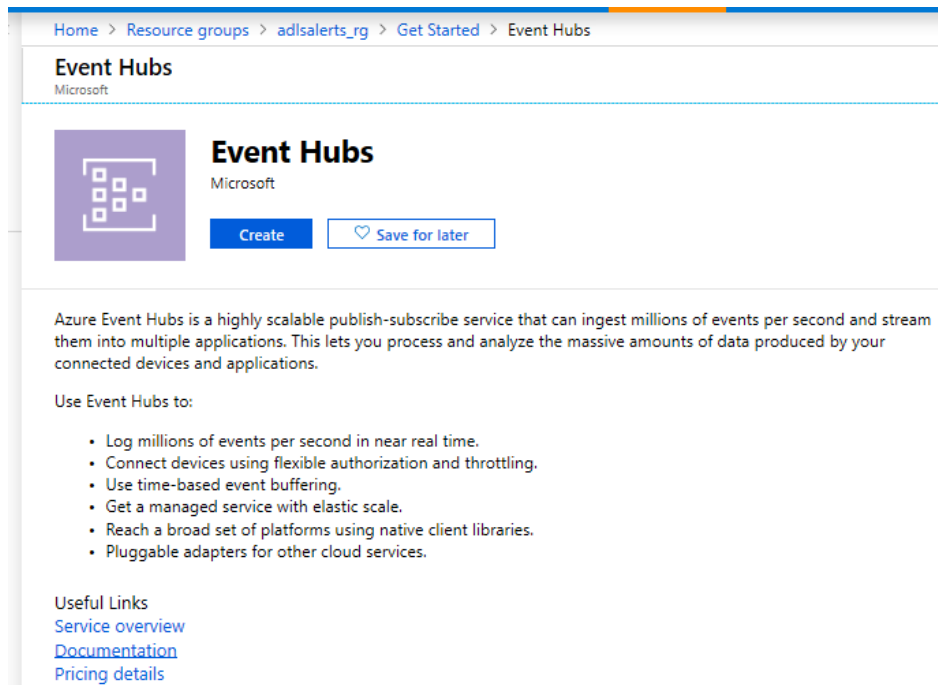


Figure 17: Event Hubs

Click on create action button in the interface, like the one shown in the above visual. Fill in the necessary configuration as in the visual below:



Create Namespace

Event Hubs

* Name

adlsngen1alerts2ehns



.servicebus.windows.net

* Pricing tier ([View full pricing details](#))

Basic (1 Consumer group, 100 Brokered co... ▾



Make this namespace zone redundant ⓘ

* Subscription

Data Migration Jumpstart POD1 ▾

* Resource group

adlsalerts_rg ▾

[Create new](#)

* Location

East US 2 ▾

* Throughput Units



1



Enable Auto-Inflate ⓘ

Create

Figure 18: Create Event Hub Namespace

Upon successful creation of the event hub, you can check the event hub resource from the following Resource Group overview page as in the following visual:

Home > Resource groups > adlsalerts_rg

adlsalerts_rg
Resource group

Search (Ctrl+/)

Overview

- Activity log
- Access control (IAM)
- Tags
- Events

Settings

- Quickstart
- Deployments
- Policies
- Properties

Subscription (change)
Data Migration Jumpstart POD1

Deployments
9 Succeeded

Subscription ID
aa841837-2566-4774-b4e5-cbd2531bc832

Tags (change)
[Click here to add tags](#)

Filter by name... All types All locations No grouping

7 items ☐ Show hidden types

NAME	TYPE	LOCATION
adlsalertsgen1	Data Lake Storage Gen1	East US 2
adlsngen1alerts2ehns	Event Hubs Namespace	East US 2

Figure 19: Resources Overview - Event Hub Namespace Listing

Once you have the Event Hub Namespace defined, the next step is to define event hub where our events from the ADLS gen1 can be captured. Here's the reference visual to help you define an Event Hub, under this newly defined Event Hub Namespace:

Home > Resource groups > adlsalerts_rg > adlsngen1alerts2ehns > Create Event Hub

Create Event Hub

Event Hubs

* Name *i*

adlsngen1alertseventhubs ✓

Partition Count *i*

2

Message Retention *i*

1

Capture *i*

On Off

Create

Figure 20: Create Event Hub

4.2 Configure ADLS Diagnostics

Now that we have an Event Hub name space and Event Hub defined, let's configure diagnostics to also stream events to the event hub. Here's the visual for a quick reference:

Home > Resource groups > adlsalerts_rg - Diagnostic settings > Diagnostics settings

Diagnostics settings

Save Discard Delete

Name
adlsgen1diagnostics2sa

☒ Archive to a storage account

Storage account
adlsgen1alertssa

☒ Stream to an event hub

Event hub
Configure

☒ Send to Log Analytics

Subscription
Data Migration Jumpstart POD1

Log Analytics Workspace
adlsgen1alertsLAWS (eastus2)

LOG

☒ Audit Retention (days) 1

☒ Requests Retention (days) 1

Select event hub

Subscription
Data Migration Jumpstart POD1

* Select event hub namespace
adlsgen1alerts2ehns

Select event hub name (optional) ⓘ
adlsgen1alertseventhub

* Select event hub policy name
RootManageSharedAccessKey

OK

Now that we have an Event Hub namespace defined, let's proceed to define a diagnostic setting to point ADLS events to the newly created event hub. Navigate the Azure Portal Path - Home → Monitor – Diagnostics Settings and add a new diagnostic setting. Here's the visual for reference:

Diagnostics settings

Save Discard Delete

* Name
adlsgen1diagnostics2eventhub

☐ Archive to a storage account

☒ Stream to an event hub

Event hub
adlsgen1alerts2eventhub (RootManageSharedAccessKey)

☐ Send to Log Analytics

Select event hub

Subscription
Data Migration Jumpstart POD1

* Select event hub namespace
adlsgen1alerts2eventhub

Select event hub name (optional) ⓘ
Select event hub name

* Select event hub policy name
RootManageSharedAccessKey

Figure 21: Selecting Event Hub Namespace

As you observe from the visual above, this time we are selecting the option to stream events to an event hub. Once you have selected the corresponding event hub, on the diagnostic settings also select the log options – Requests, and for metrics select All Metrics. Hit save on the top navigation pane (this is enabled once you select what to capture i.e., logs and metrics).

Here's the overview page where both the diagnostic settings (log analytics & event hub) are listed:

Home > Monitor - Diagnostics settings

Monitor - Diagnostics settings

Search (Ctrl+J) Refresh

Subscription: Data Migration Jumpstart POD1 Resource group: adlsgen1_rg Resource type: Data Lake Storage Gen1 Resource: adlsgen1

Data Migration Jumpstart POD1 > adlsgen1_rg > adlsgen1

NAME	STORAGE ACCOUNT	EVENT HUB	LOG ANALYTIC	EDIT SETTING
adlsgen1diagnostics2eventhub	-	adlsgen1alerts2eventhub	-	Edit setting
adlsgen1diagnostics2sa	adlsgen1alertssa	-	adlsgen1alertslaws	Edit setting

[+ Add diagnostic setting](#)

Figure 22: Diagnostic Settings Overview

4.3 Define Azure Function to filter ADLS events

Once we have configured diagnostics to log events to the event hub, we can proceed to defining consumer applications to filter and raise alerts using Microsoft's Event Hub SDK (custom application). The other option here is to define Logic Apps. In this section, we will walk through the necessary steps to define an Azure Function as a trigger that is applied whenever an event is logged into the Event Hub.

The very first step is to define a Function App. This can be accomplished by choosing the Function App tab under the left navigation pane – favorites tab, in your Azure Portal.

Home > Function App > Function App

Function App Create

* App name
adlsgen1alertsDotNetFnApp ✓
.azurewebsites.net

* Subscription
Data Migration Jumpstart POD1

* Resource Group ⓘ
☐ Create new ☒ Use existing
adlsalerts_rg

* OS
☒ Windows ☐ Linux

* Hosting Plan ⓘ
Consumption Plan

* Location
East US 2

* Runtime Stack
.NET

* Storage ⓘ
☐ Create new ☒ Use existing
adlsgen1alertssa

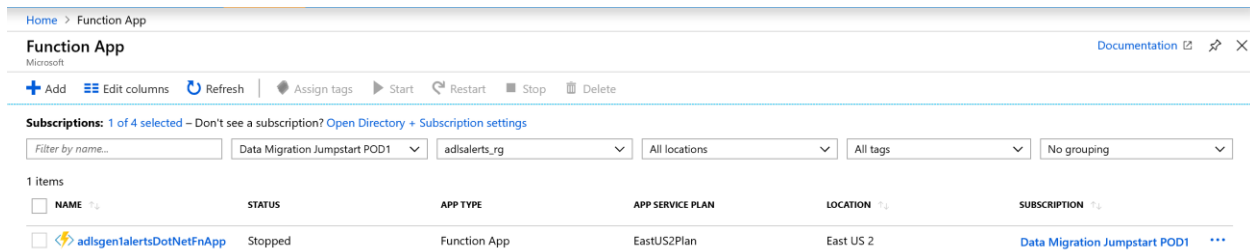
Application Insights
adlsgen1alertsDotNetFnApp

For optimal performance you should use a storage account in the same region as the Function App.

Create Automation options

Figure 23: Create Function App

You can verify the created resource – function app, from the function apps overview blade (view):

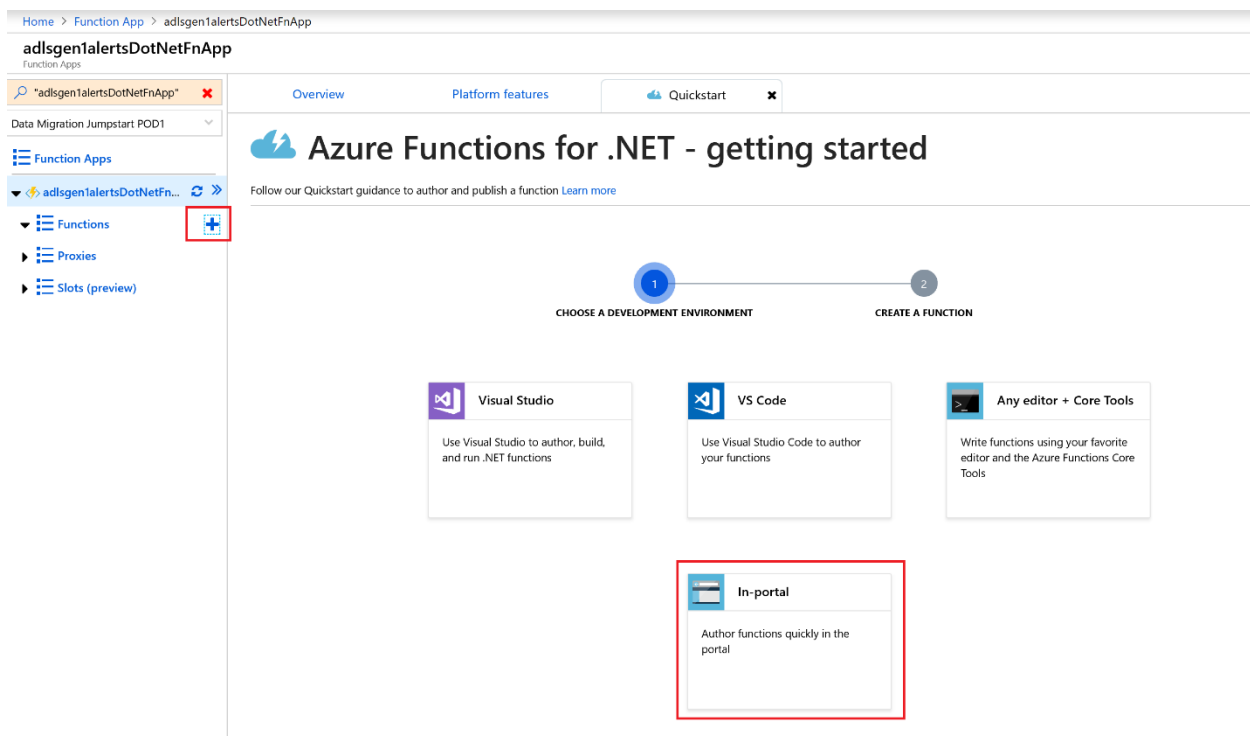


The screenshot shows the 'Function App' overview blade in the Azure portal. It includes a table with one function app listed. The table has columns for NAME, STATUS, APP TYPE, APP SERVICE PLAN, LOCATION, and SUBSCRIPTION. The function app 'adlsgen1alertsDotNetFnApp' is shown with a status of 'Stopped'.

NAME	STATUS	APP TYPE	APP SERVICE PLAN	LOCATION	SUBSCRIPTION
adlsgen1alertsDotNetFnApp	Stopped	Function App	EastUS2Plan	East US 2	Data Migration Jumpstart POD1

Figure 24: Existing Function Apps Overview

Subsequently will define an in-portal function using one of the available templates as shown in the following visual:



The screenshot shows the 'Azure Functions for .NET - getting started' page. It features a progress bar with two steps: 'CHOOSE A DEVELOPMENT ENVIRONMENT' and 'CREATE A FUNCTION'. Under 'CHOOSE A DEVELOPMENT ENVIRONMENT', there are three options: 'Visual Studio', 'VS Code', and 'Any editor + Core Tools'. Under 'CREATE A FUNCTION', there is one option: 'In-portal'. The 'In-portal' option is highlighted with a red box.

Figure 25: Define Azure Function (In-portal template)

Then, select more templates...

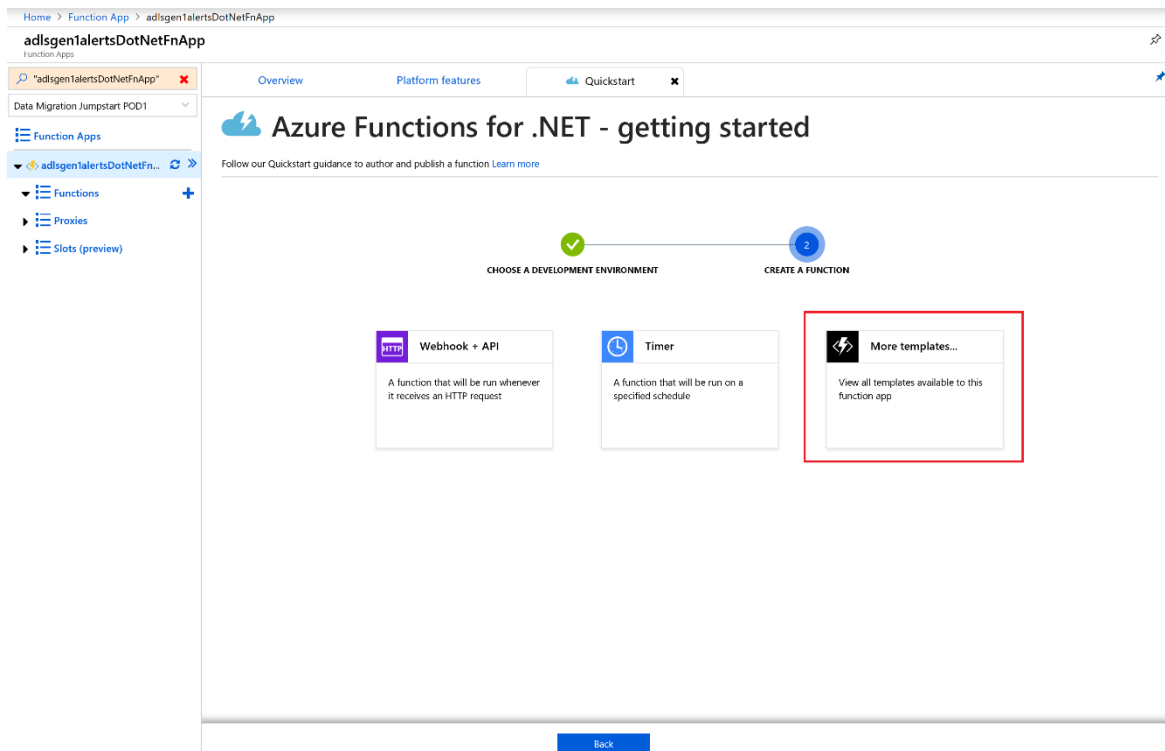


Figure 26: Azure Function - More Templates...

The next step is to search for Azure Event Hub Trigger template as shown below:

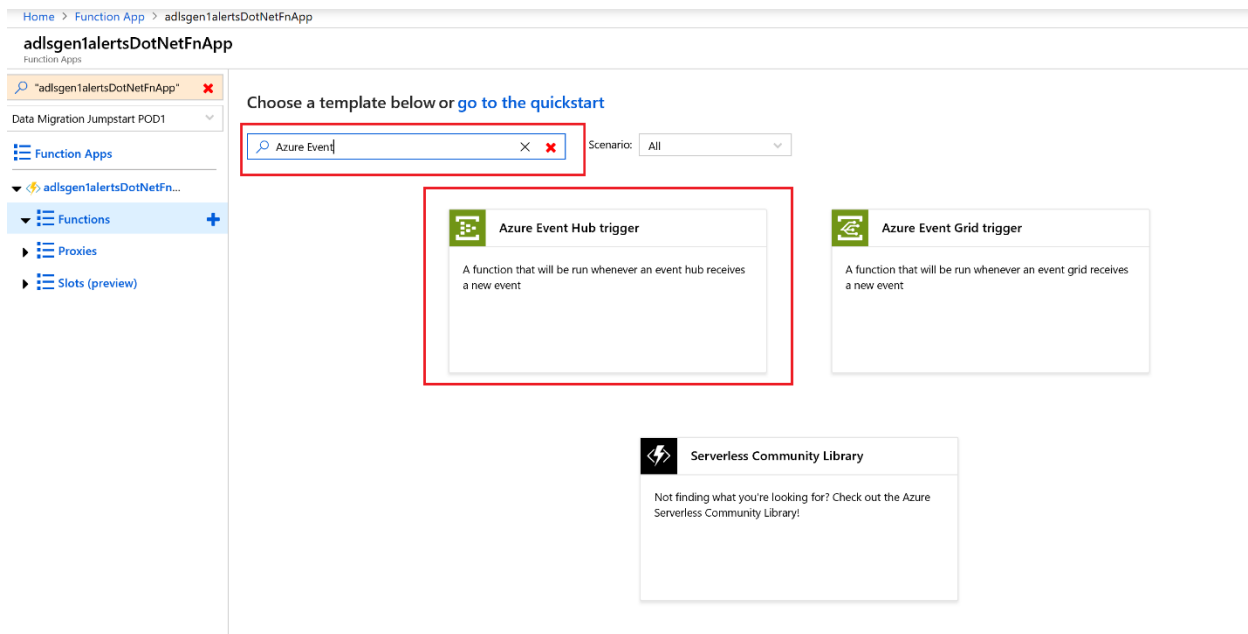


Figure 27: Azure Functions - Azure Event Hub Trigger

Install the missing plugin if prompted,

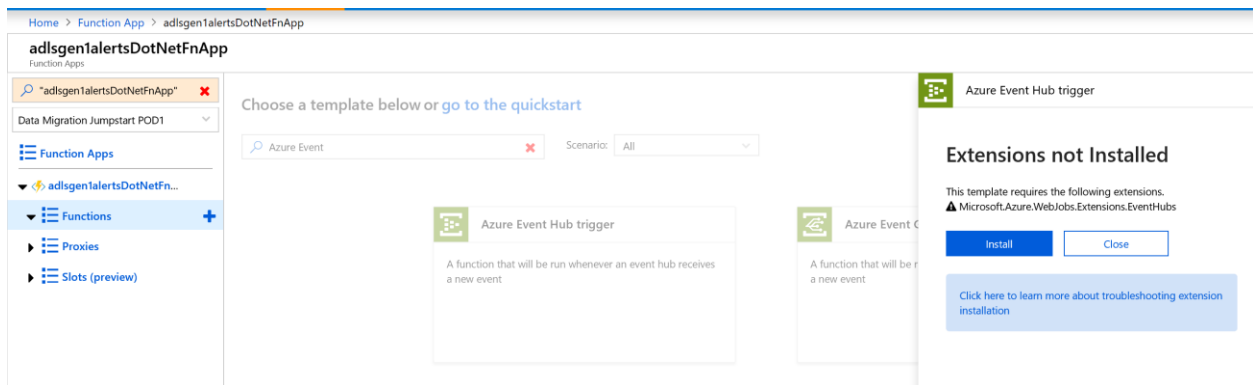
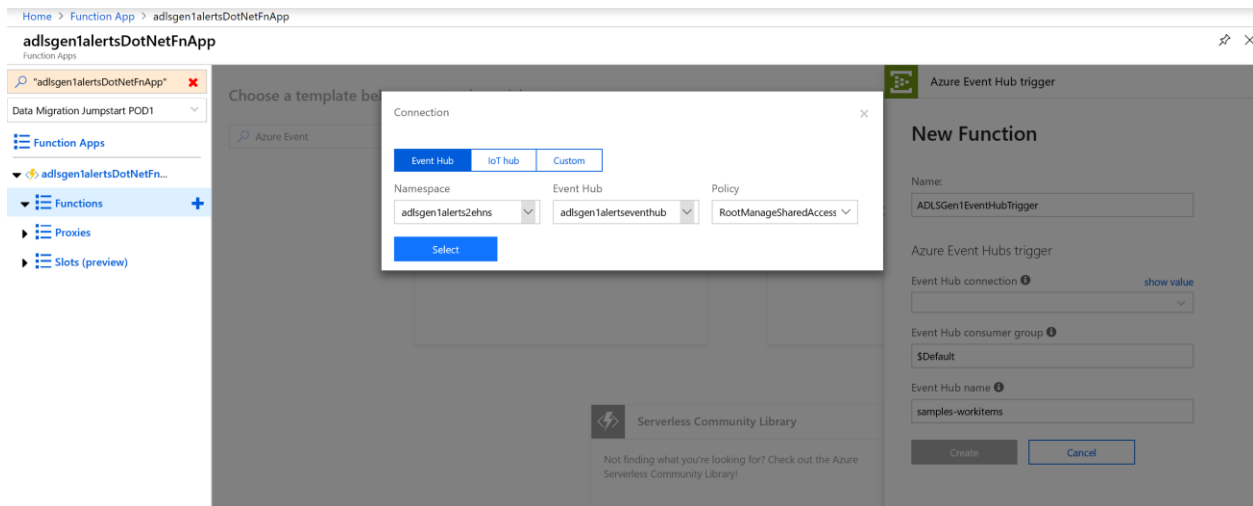


Figure 28: Microsoft.Azure.WebJobs.Extensions.EventHubs Extension

Select the available event hub namespace, event hub and access policy from your subscription.



Click create and a sample code will be generated by the designer as shown below.

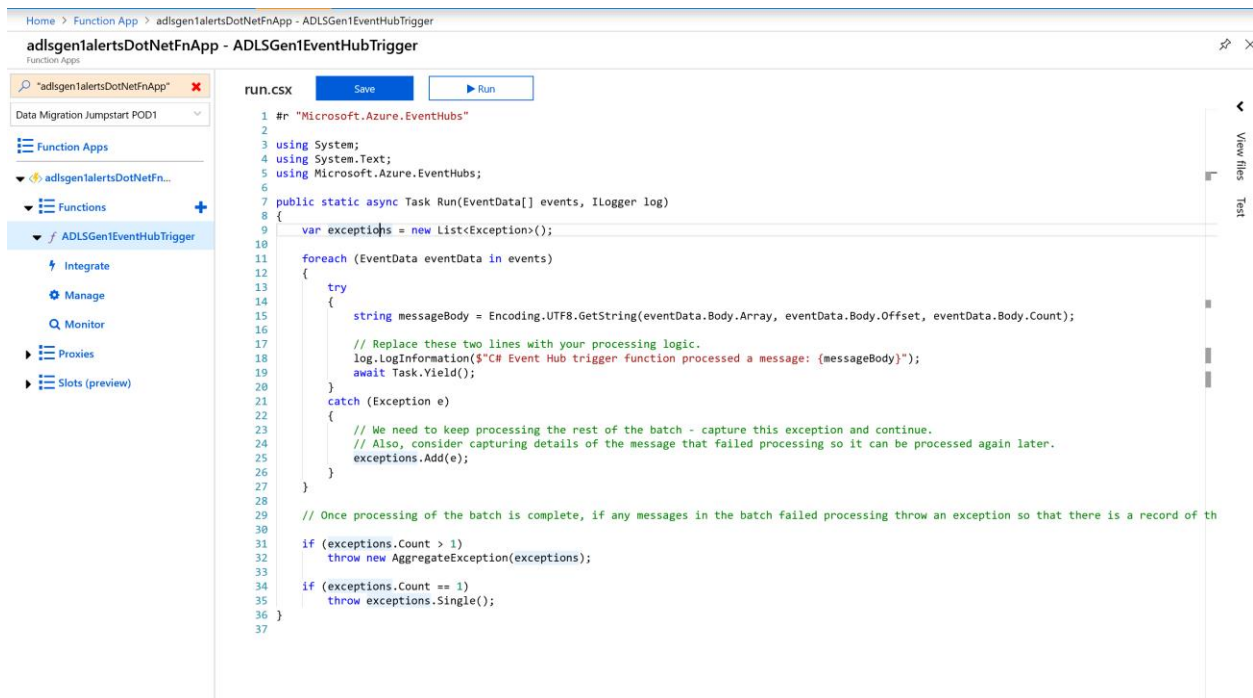


Figure 29: Sample Code generated by Designer

Optionally , we can also enhance the sample code, for example to send the events to downstream applications to consume the events.

```
private static void SendEventDownStream(string message) {
    var httpRequest =
        (HttpRequest)WebRequest
            .Create("https://webhook.site/a42e189a-0854-40a4-b629-dc838e750088");
    httpRequest.ContentType = "application/json";
    httpRequest.Method = "POST";

    using (var streamWriter = new StreamWriter(httpRequest.GetRequestStream())) {
        streamWriter.Write(message);
        streamWriter.Flush();
        streamWriter.Close();
    }

    var httpResponse = (HttpWebResponse)httpRequest.GetResponse();
    using (var streamReader = new StreamReader(httpResponse.GetResponseStream())) {
        var result = streamReader.ReadToEnd();
    }
}
```

Figure 30: Generated Code customization sample

5 ADLS Event Notification Alerts using Event Grid

The third and last approach discussed in this white paper is the approach where Event Grid is used to setup event notification alerts. This is more of a futuristic feature, where events from Data lake can be directly ingested into Event Grid. Here is the empirical view of the event flows:

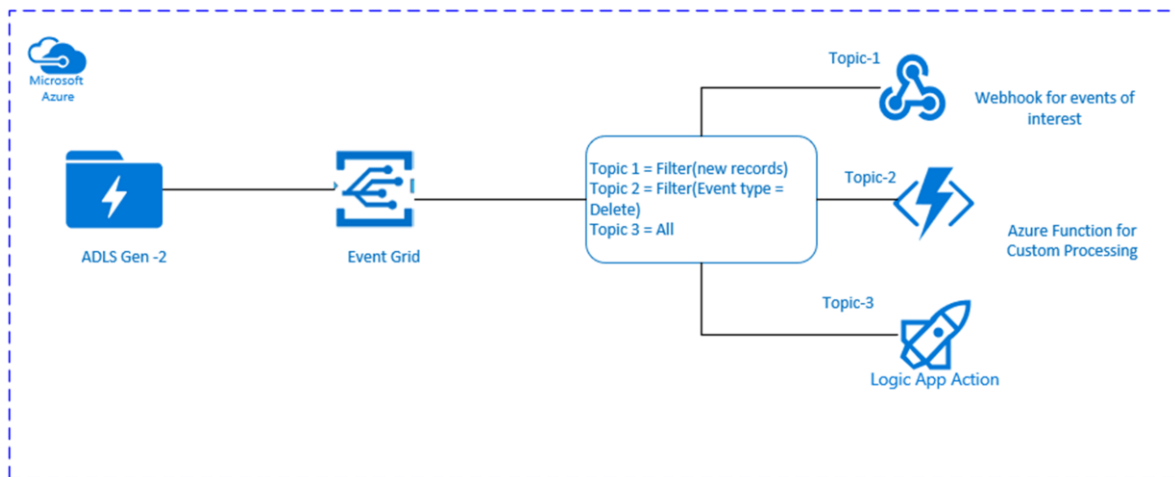


Figure 31: Event Notification Alerts using Event Grid

5 Appendices

5.1 Appendix - External References

- [Struggling to get insights for your Azure Data Lake Store? Azure Log Analytics can help!](#)
- [Azure Log Analytics](#)
- [Azure Monitor](#)
- [Azure Event Hubs](#)
- [Azure Logic Apps](#)

6 Feedback and suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Data SQL Ninja Team (datasqlninja@microsoft.com). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the [Azure Database Migration Guide](#).