

ARM Overview


Last updated by | Charlene Wang | Jan 3, 2023 at 12:30 AM PST

Created On: Nov 17, 2022
Authored by: luyang1
Reviewed By: zhizhwan

Contents



- [What is ARM?](#)
 - [ARM Data for Sync Operation](#)
 - [ARM Data for Async Operation](#)
- [Kusto tables:](#)
- [Kusto Sample queries](#)
- [Reference:](#)

What is ARM?


[Azure Resource Manager](#)  is the deployment and management service for Azure. It provides a management layer that enables you to create, update, and delete resources in your Azure account. You use management features, like access control, locks, and tags, to secure and organize your resources after deployment. ARM accepts calls from Azure Portal, Powershell/CLI, REST API, SDKs..., and routes requests to proper Resource Provider.

[What is Azure Resource Manager?](#) 

[What are ARM templates?](#) 


We've introduced a new language named [Bicep](#)  that offers the same capabilities as ARM templates but with a syntax that's easier to use. Each Bicep file is automatically converted to an ARM template during deployment. If you're considering infrastructure as code options, we recommend looking at Bicep. For more information, see [What is Bicep?](#) .

ARM Data for Sync Operation

A Synchronous operation or Sync operation refers to the lifecycle of a single http request, during which ARM fulfills the request by running its internal operations and interacting with target resource providers. [ARM data for Sync operations](#)  records details about customers, ARM, RPs, and their interactions during the lifecycle of http requests.

ARM Data for Async Operation

An Asynchronous operation or Async operation is triggered when either ARM or the target resource provider needs to fulfil a customer request in an asynchronous way, during which ARM runs jobs to track and manage operation status for all parties. In some cases, the execution of an Async operation also triggers follow-up http requests and operations to fulfill the root request.

A common, successful Azure Async Operation includes a series of steps marked by 6 milestones, refer to [Async Operation](#) 

T1: A customer request arrives at ARM front door.
 T2: ARM forwards the request to the target RP.
 T3: The RP sends a response to ARM with a long operation header.
 T4: ARM creates a long operation job internally and sends a response to the customer that the request is accepted.
 T5: The RP completes its long operation.
 T6: ARM detects the completion of the RP operation, validates the resource status, and then closes the Async operation.

Also see: [Client -> ARM -> RP flow graph](#)

Kusto tables:

ARM cluster name: armprod

HttpIncomingRequests:
 contains all http requests coming into ARM front door layer (REST API, CLI, Portal, ARM Template...) This will show the high-level activities for a correlation ID.

HttpOutgoingRequests:
 contains all http requests that ARM sends to RPs.
 If a failure (ie. HTTP 429 or HTTP 500) is being returned from a resource provider then the root cause of the failure is shown in the details.

EventServiceEntries:
 Activity Logs within ARM; summary of operations and errors. This will show the high-level activities for a correlation ID.

ShoeboxEntries:
 Logs written to the customer's activity logs. This includes the error message returned to the customer.

Traces:
 Internal operation flow for ARM. Includes items such as RBAC and Policy evaluations.

Errors:
 Internal ARM errors. Useful if the failure is in the ARM layer rather than in a resource provider. Will often show the root cause of the failure.

Deployments:
 High-level records to declarative deployment status (success/ failure) and summary of resources involved

DeploymentOperations:
 Detailed information aligned with Deployments table. One row representing one resource and can be linked to operations.

ProviderTraces:
 Operations within resource provider. Some resource providers will log errors here, but most don't.

Kusto Sample queries

```
// Check ARM layer log/error - union three tables: HttpIncomingRequests, HttpOutgoingRequests, EventServiceEnt
let server = '{serverName}';
let db = '{dbName}';
let subID = '{SubscriptionId}';
let crlID = '{correlationId}';
let startTime = ago(1d);
let endTime = now();
union withsource=SourceTable kind=outer HttpIncomingRequests,HttpOutgoingRequests,EventServiceEntries
| where TIMESTAMP >= startTime and TIMESTAMP <= endTime
| where subscriptionId =~ subID
| where properties contains server //and properties contains db
| where correlationId =~ crlID //if the correlationId of the operation is unknown then remove this filter fir
| where isempty(server) or resourceUri endswith strcat('/',server) or resourceUri has strcat('/',server, '/')
| extend statusMessage = parse_json(tostring(parse_json(properties).statusMessage))
| extend statusMessageStatus = statusMessage.status
| extend error = statusMessage.error
| extend errorMessage = statusMessage.error.message
| extend errorMessageCode = tostring(statusMessage.error.details[0].code)
| extend errorMessageDetails = tostring(statusMessage.error.details[0].message)
| where errorMessageDetails != 'The operation timed out and automatically rolled back. Please retry the operat
| parse resourceUri with '/subscriptions/' subId '/resourcegroups/' RGname 'providers/Microsoft.Sql/' Resourc
| parse resourceUri with '/subscriptions/' subId2 '/resourceGroups/' RGname2 'providers/Microsoft.Sql/' Resou
| extend Message = iif(errorMessageDetails!=', errorMessageDetails, errorMessage)
| extend Resource = iif(isnotempty(Resource), Resource, Resource2)
| project SourceTable, PreciseTimeStamp, resourceProvider, operationName, Resource
, status, subStatus, errorCode, errorMessage, Message, statusMessageStatus, errorMessageCode
, eventTimestamp, correlationId, operationId, resourceUri, targetResourceProvider, targetUri
, properties, authorization, claims, Deployment, principal0id, principalPuid

// authorization column contains: scope, action, role info, etc
// claims column contains: appid, ipaddr, user name(operation committer), user email account info(sometimes te

HttpIncomingRequests
| where TIMESTAMP > {startdate} and TIMESTAMP < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where correlationId =~ '{correlationId}'
| where targetUri contains '{servername}' and targetUri contains '{dbname}'
//| where httpMethod == "PATCH" //scale up -> PATCH
| project-reorder httpStatusCode, targetUri
| summarize count() by TIMESTAMP, httpStatusCode, TaskName, targetResourceProvider, targetResourceType, httpMe
, operationName, targetUri, userAgent, clientIpAddress, RoleLocation, subscriptionId, additionalProperties

HttpOutgoingRequests
| where TIMESTAMP > {startdate} and TIMESTAMP < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where correlationId =~ '{correlationId}'
| where targetUri contains '{servername}'
| where targetUri contains '{databasename}'
| project TIMESTAMP, httpStatusCode, TaskName, targetResourceProvider, targetResourceType, correlationId, oper
subscriptionId, httpMethod, RoleLocation, additionalProperties

EventServiceEntries
| where PreciseTimeStamp > {startdate} and PreciseTimeStamp < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where resourceProvider =~ 'Microsoft.Sql'
| where eventCategory != 'Policy'
| where isempty('{servername}') or resourceUri endswith strcat('/', '{servername}') or resourceUri has strcat('
| extend statusMessage = parse_json(tostring(parse_json(properties).statusMessage))
| extend statusMessageStatus = statusMessage.status
| extend error = statusMessage.error
| extend errorMessage = statusMessage.error.message
| extend errorMessageCode = tostring(statusMessage.error.details[0].code)
| extend errorMessageDetails = tostring(statusMessage.error.details[0].message)
| where errorMessageDetails != 'The operation timed out and automatically rolled back. Please retry the operat
| parse resourceUri with '/subscriptions/' subId '/resourcegroups/' RGname 'providers/Microsoft.Sql/' Resourc
//| parse resourceUri with '/subscriptions/' subId2 '/resourceGroups/' RGname2 'providers/Microsoft.Sql/' Res
| parse operationName with 'Microsoft.Sql/' Operation
```

```

| extend Message = iif(errorMessageDetails!='', errorMessageDetails, errorMessage)
| extend Resource = iif(isnotempty(Resource), Resource, Resource2)
| project PreciseTimeStamp, eventTimestamp, correlationId, eventName, status, subStatus, Operation, Resource,
| order by PreciseTimeStamp asc nulls last

ShoeboxEntries
| where TIMESTAMP > {startdate} and TIMESTAMP < {enddate}
| where resourceId contains '{servername}'
| project PreciseTimeStamp, correlationId, operationName, category, resultType, resultSignature, resultDescrip
, ActivityId, ProviderGuid, Deployment

DeploymentOperations
| where PreciseTimeStamp > {startdate} and PreciseTimeStamp < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where correlationId =~ '{correlationId}'
| where executionStatus == "Failed"

Deployments
| where PreciseTimeStamp > {startdate} and PreciseTimeStamp < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where correlationId =~ '{correlationId}'

JobTraces
| where TIMESTAMP > {startdate} and TIMESTAMP < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where correlationId =~ '{correlationId}'
| project PreciseTimeStamp, Deployment, RoleInstance, TaskName, operationName, ActivityId, subscriptionId, cor
, jobPartition, message, jobId, additionalProperties, principalOid, exception

Traces
| where TIMESTAMP > {startdate} and TIMESTAMP < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where correlationId =~ '{correlationId}'
//| where message contains '{servername}'
| project PreciseTimeStamp, TaskName, correlationId, operationName, message, exception, SourceNamespace, addit
, ActivityId, subscriptionId, tenantId

Errors
| where TIMESTAMP > {startdate} and TIMESTAMP < {enddate}
| where tenantId =~ '{tenantId}'
| where correlationId =~ '{correlationId}'
| project PreciseTimeStamp, correlationId, operationName, message, exception, additionalProperties, tenantId,

ProviderTraces
| where TIMESTAMP > {startdate} and TIMESTAMP < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where correlationId =~ '{correlationId}'
| project PreciseTimeStamp, correlationId, operationName, message, exception, ActivityId, principalOid, provid

ProviderErrors
| where TIMESTAMP > {startdate} and TIMESTAMP < {enddate}
| where subscriptionId =~ '{subscriptionId}'
| where correlationId =~ '{correlationId}'
| project PreciseTimeStamp, correlationId, providerNamespace, resourceType, operationName, message, exception,

```



Reference:

ARM wiki: [Basic ARM Data knowledge](#) 

How good have you found this content?

