Provisioning operations breakdown - single Kusto query

Last updated by | Francisco Javier Pardillo Martin | Aug 28, 2022 at 3:33 AM PDT

Contents

Request

Based in Troubleshoot PostgreSQL Flexible server Provisioning Issues to create a single query to get all the relevant information.

The aim of this query is to improve the time to collect all the required information from engineers' perspective.

Investigation/Analysis

We´ll try to provide a single Kusto query to get all the details of a provisioning issue by only requesting the Timestamp, Subscription ID, Correlation ID or ServiceRequestID (Tracking ID from the customer Error message).

It´s important to try to filter also by Timestamp range, even if we have the ServiceRequestID, to avoid vast number of rows scans, so improving the global query performance.

Query should be run from ARMProd cluster, database ARMProd.

```
let GlobalQuery = (tab:string, clust:string) {union isfuzzy=true
  (cluster('sqlazureau2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='austr
, (cluster('sqlazureau2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='austr
, (cluster('sqlazureau2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='austr
, (cluster('sqlazureau2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='austr
, (cluster('sqlazurebr2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='brazi
, (cluster('sqlazurebr2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='brazi
, (cluster('sqlazureneu2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='nort
, (cluster('sqlazureca2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='canad
, (cluster('sqlazureca2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='canad
, (cluster('sqlazurecus2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='cent
, (cluster('sqlazurecus2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='cent
, (cluster('sqlazureeas2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='east
, (cluster('sqlazureeus12.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='eas
, (cluster('sqlazureeus22.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='eas
, (cluster('sqlazureeus22.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='eas
, (cluster('sqlazurefra.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='franc
, (cluster('sqlazurefra.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='franc
, (cluster('sqlazureince2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='cen
, (cluster('sqlazureinso2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='sou
, (cluster('sqlazureinwe2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='wes
, (cluster('sqlazureja2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='japan
, (cluster('sqlazureja2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='japan
, (cluster('sqlazurekor.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='korea
, (cluster('sqlazurekor.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='korea
, (cluster('sqlazurencus3.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='nor
, (cluster('sqlazurescus2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='sou
, (cluster('sqlazureseas2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='sou
, (cluster('sqlazuresouthafrica.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust
, (cluster('sqlazuresouthafrica.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust
, (cluster('sqlazureuk2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='ukwes
, (cluster('sqlazureuk2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='uksou
, (cluster('sqlazrwcus.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='westce
, (cluster('sqlazureweu2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='west
, (cluster('sqlazurewus1.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='west
, (cluster('sqlazurewus2.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='west
, (cluster('sqlazureusw3.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='west
, (cluster('sqlazureuae.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='uaeno
, (cluster('sqlazureswitzerland.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust
, (cluster('sqlazureswitzerland.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust
, (cluster('sqlazureses.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='swede
, (cluster('sqlazuresec.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='swede
, (cluster('sqlazurenorway.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='no
, (cluster('sqlazurenorway.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='no
, (cluster('sqlazuregermany.kustomfa.windows.net').database('sqlazure1').table(tab)|extend allow=iif(clust=='g
|where allow==1};
let operation=materialize (HttpOutgoingRequests
|where TIMESTAMP > datetime(2022-08-18 11:00:00) and TIMESTAMP < datetime(2022-08-18 11:10:00)
//| where subscriptionId =~ '2eefea3d-c80a-46e4-a856-cd8aefad60fe'
//| where correlationId =~ 'b1b44599-9443-4438-803b-9ae298
| where serviceRequestId =~ 'ecfcfbbb-0423-4006-a3a8-2a0819d6f597'
| where operationName contains "AZUREASYNCOPERATION"
| extend location = extract('locations/([a-z|A-Z|0-9|]+)', 1, targetUri, typeof(string))
| extend asyncId = extract('azureAsyncOperation/([a-z|A-Z|0-9|-]+)', 1, targetUri, typeof(string))
| where isnotempty( asyncId)
| project TIMESTAMP, correlationId, asyncId, serviceRequestId,TaskName, location, subscriptionId, operationNam
);
let clust=toscalar(operation|order by TIMESTAMP desc |limit 1|summarize max(location));
let asyncId=toscalar(operation|order by TIMESTAMP desc| limit 1|summarize max(asyncId));
let correlationId=toscalar(operation|order by TIMESTAMP desc| limit 1|summarize max(correlationId));
// Operation details
operation;
let GlobalMonOrcasBreadthRp = materialize(GlobalQuery('MonOrcasBreadthRp',clust));
let GlobalMonOrcasBreadthRpExceptions = materialize(GlobalQuery('MonOrcasBreadthRpExceptions',clust));
let GlobalMonOrcasBreadthResourceProvider = materialize(GlobalQuery('MonOrcasBreadthResourceProvider',clust));
// Details of operation
GlobalMonOrcasBreadthRp
| where request_id =~ asyncId
```

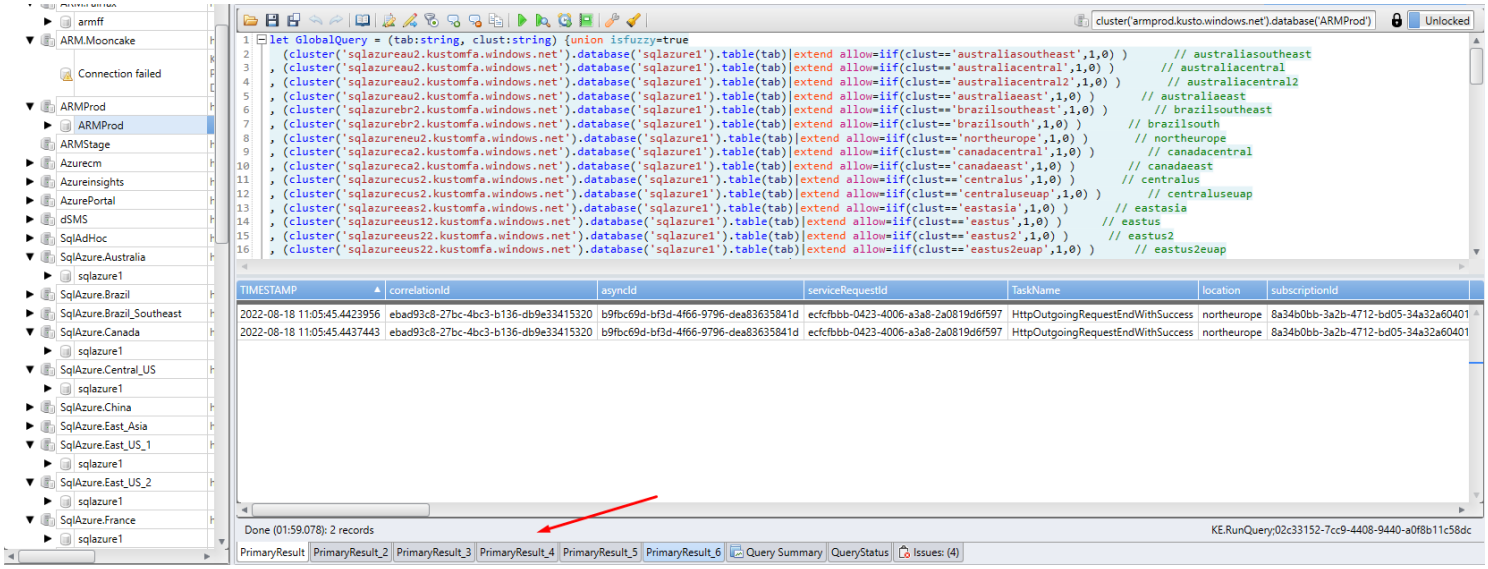
```
| where operation_parameters !=''
| project TIMESTAMP, operation_parameters;
//Get the error messages
GlobalMonOrcasBreadthRp
| where request_id =~ asyncId
| where (event contains "cancel" or event contains "timeout" or event contains "fail") or isnotempty (message)
| project TIMESTAMP, event, exception_type, stack_trace, message, operation_type, error_code, error_severity,
//Get the state and Error messages
GlobalMonOrcasBreadthRp
| where request_id =~ asyncId
| project TIMESTAMP, old_state, new_state, state,  message, error_message, error_number | order by TIMESTAMP a
//Get the state and Error messages
GlobalMonOrcasBreadthResourceProvider
| where correlation_id =~ correlationId
| project TIMESTAMP, originalEventTimestamp, elapsed_time_milliseconds, caller_address, client_routing_id, cor
//Get the Error messages detailed
GlobalMonOrcasBreadthRpExceptions
| where request_id =~ asyncId
| project TIMESTAMP, event, exception_type, message, stack_trace, level
```

Features

- The query is based in a user defined function "GlobalQuery" with 2 parameters: table name and clustername, based on clustername the query will request data only for that cluster (not for all clusters), this can be checked by using the "QueryStatus" when you execute the query and you will observe traffic only generated into the target cluster, something like this:

```
"cross_cluster_resource_usage": {
                "https://sqlazureneu2.kustomfa.windows.net/": {
                        "cache": {
                                "memory": {
                                        "hits": 41651,
                                        "misses": 2289,
                                        "total": 43940
                                },
                                "disk": {
                                        "hits": 1646,
                                        "misses": 0,
                                        "total": 1646
                                },
                                "shards": {
                                        "hot": {
                                                "hitbytes": 0,
                                                "missbytes": 0,
                                                "retrievebytes": 0
                                        },
                                        "cold": {
                                                "hitbytes": 0,
                                                "missbytes": 0,
                                                "retrievebytes": 0
                                        },
                                        "bypassbytes": 0
                                }
                        },
                        "cpu": {
                                "user": "00:00:15.7656250",
                                "kernel": "00:00:00.6406250",
                                "total cpu": "00:00:16.4062500"
                        },
                        "memory": {
                                "peak_per_node": 18373344
                        },
                        "network": {
                                "inter_cluster_total_bytes": 6050129,
                                "cross_cluster_total_bytes": 0
                        }
                }
        }
```


- The query will consider the latest asyncId found based in the initial query parameters (Timestamp, Subscription ID, Correlation ID or Timestamp, ServiceRequestID).
- There is no need to change between clusters to collect detailed operation information.
- You don´t need to change or copy the asyncId between different queries, those values are passed between subqueries by using the let (scalar) operator.
- After executing the query you can review the results for each of its subqueries by inspecting in kusto explorer the bottom tabs (PrimaryResult, PrimaryResult_2, ...) each of them belonging to a subselect in the main query.



- You can click on kusto explorer in "Query and Results to Clipboard" so, the complete query and all the subselect results will be copied in clipboard directly, no need to multiple clicks.
- This query can be used as a base query for multiple other scenarios.

Internal Reference

Troubleshoot PostgreSQL Flexible server Provisioning Issues

How good have you found this content?