Sync or Rehydrate ARM Cache

Last updated by | Daniel Valero | Apr 20, 2022 at 11:27 AM PDT

This TSG is part of GT , please contact EEE haaqel@microsoft.com for any updates.

Contents

Issue

Some customers are complaining suddenly that they are not able to see resources on Azure portal, or they can see with wrong information, especially when they are restoring their dropped resources, on that case we need to re-hydrate ARM Cache as below using Jarvis:

Security considerations

The below Jarvis instructions can be run only on a SAVM machines only, as SAW-enforced resources, such as Jarvis and JIT are allowed only on a SAVM (SAVM (formerly known internally as vSAW) which allows CSS personnel to use their standard corporate machines to access production websites, via cloud-based Secured Access Virtual Machines (SAVMs). This is not a replacement for SAW, but rather a more secure variation of the exception group.

If you are not a member of CSS SAVM program, use Eligibility & Joining - STRIKE Central (strikecommunity.azurewebsites.net)

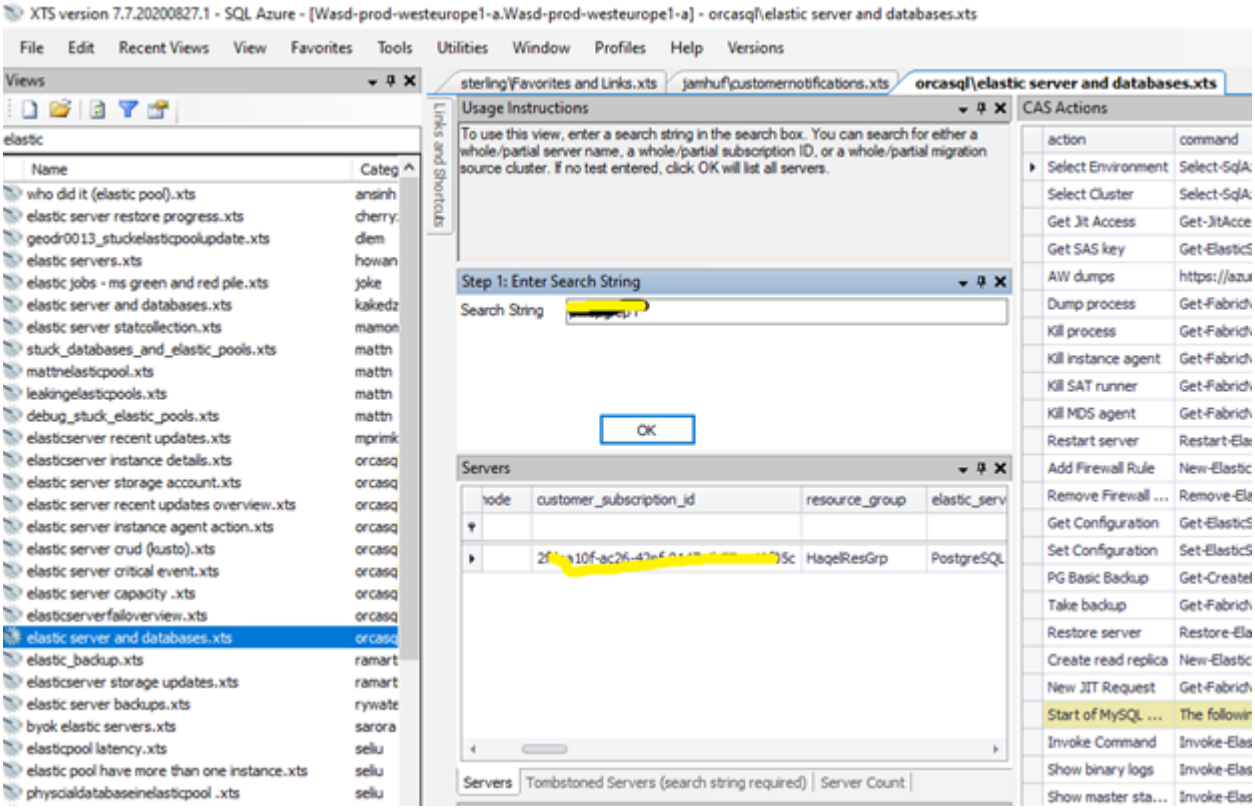
For how to use and create your own SAVM, go too Using SAVM - STRIKE Central (strikecommunity.azurewebsites.net)

Mitigation

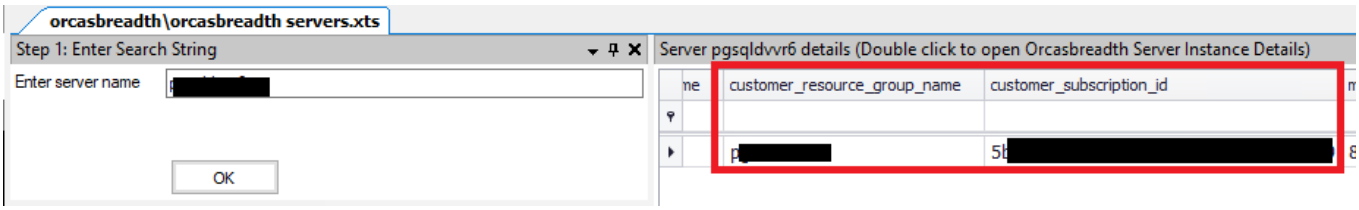
1. Find and verify server info like subscription ID, resource group, and region.

   If you know the region and the server name, You can use XTS views:

   - For Single Server use *orcasql\elastic server and databases.xts*



- For Flexible Server Postgresql use *orcasbreadth\orcasbreadth servers.xts*



You can use below KUSTO query to check needed info:

- For Single Server

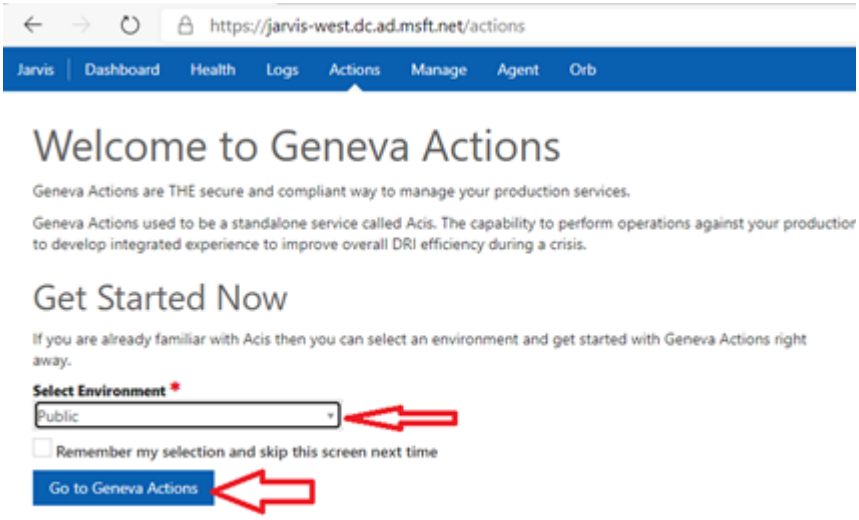
```
let GlobalMonAnalyticsElasticServersSnapshot = union
cluster('sqlazureau2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot
, cluster('sqlazurebr2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsho
, cluster('sqlazureca2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsho
, cluster('sqlazurecus2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsh
, cluster('sqlazureeas2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsh
, cluster('sqlazureeus12.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnaps
, cluster('sqlazureeus22.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnaps
, cluster('sqlazurefra.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsho
, cluster('sqlazureince2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnaps
, cluster('sqlazureinso2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnaps
, cluster('sqlazureinwe2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnaps
, cluster('sqlazureja2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsho
, cluster('sqlazurekor.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsho
, cluster('sqlazurencus3.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnaps
, cluster('sqlazureneu2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsh
, cluster('sqlazurescus2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnaps
, cluster('sqlazureseas2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnaps
, cluster('sqlazuresouthafrica.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServer
, cluster('sqlazureuk2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsho
, cluster('sqlazrwcus.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot
, cluster('sqlazureweu2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsh
, cluster('sqlazurewus1.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsh
, cluster('sqlazurewus2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapsh
;
GlobalMonAnalyticsElasticServersSnapshot
| where name =~ "{servername}"
| summarize arg_max(TIMESTAMP, *) by elastic_server_id
| extend IsPFS = iif(fsm_extension_data contains '<PropertyName>IsPFS</PropertyName><Value>False</Va
| project name,elastic_server_id, ['state'], dropped_date, tombstoned_to_dropdate = datetime_add('day
```

- For Flexible Server PostgreSQL

```
let AllRegionTable = (tableName:string)
{
  cluster('sqlazureau2').database('sqlazure1').table(tableName)
  | union cluster('sqlazurebr2').database('sqlazure1').table(tableName)
  | union cluster('sqlazureca2').database('sqlazure1').table(tableName)
  | union cluster('sqlazurecus2').database('sqlazure1').table(tableName)
  //| union cluster('sqlazurechi.kusto.chinacloudapi.cn').database('MoonCake').table(tableName)
  | union cluster('sqlazureeas2').database('sqlazure1').table(tableName)
  | union cluster('sqlazureeus12').database('sqlazure1').table(tableName)
  | union cluster('sqlazureeus22').database('sqlazure1').table(tableName)
  | union cluster('sqlazurefra').database('sqlazure1').table(tableName)
  //| union cluster('sqlazureger.kusto.cloudapi.de').database('BlackForest').table(tableName)
  | union cluster('sqlazureince2').database('sqlazure1').table(tableName)
  | union cluster('sqlazureja2').database('sqlazure1').table(tableName)
  | union cluster('sqlazurekor').database('sqlazure1').table(tableName)
  | union cluster('sqlazurencus3').database('sqlazure1').table(tableName)
  | union cluster('sqlazureneu2').database('sqlazure1').table(tableName)
  | union cluster('sqlazurescus2').database('sqlazure1').table(tableName)
  | union cluster('sqlazureseas2').database('sqlazure1').table(tableName)
  | union cluster('sqlazuresouthafrica').database('sqlazure1').table(tableName)
  //| union cluster('sqlazureuae').database('sqlazure1').table(tableName)
  | union cluster('sqlazureuk2').database('sqlazure1').table(tableName)
  //| union cluster('sqlazureusg.kusto.usgovcloudapi.net').database('FairFax').table(tableName)
  | union cluster('sqlazureweu2').database('sqlazure1').table(tableName)
  | union cluster('sqlazurewus1').database('sqlazure1').table(tableName)
  | union cluster('sqlazurewus2').database('sqlazure1').table(tableName)
  | union cluster('sqlazureeas2').database('sqlazure1').table(tableName)
};
AllRegionTable('MonOrcasBreadthCMSSnapshot')
//| where TIMESTAMP > <provision_start_time> and TIMESTAMP < <provision_end_time>
| where subscription_id  =~ '{}'
  or server_name =~ '{}'
| where server_state != 'Tombstoned'
| project server_name, server_type, server_version, ClusterName,  code_package_version,server_sku, st
| limit 1
```

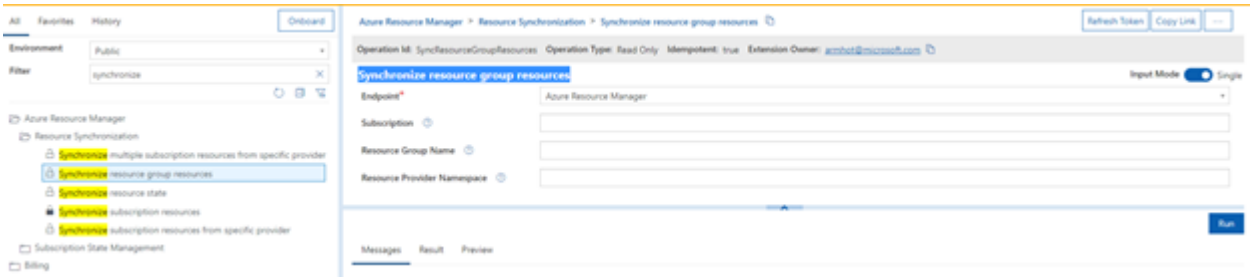
NOTE: Resource names are case sensitive in the old WCF API implementation, so make sure to preserve the correct casing for all resource names as presented in CMS for the next steps.

2. Go to Jarvis ↗ Geneva Action



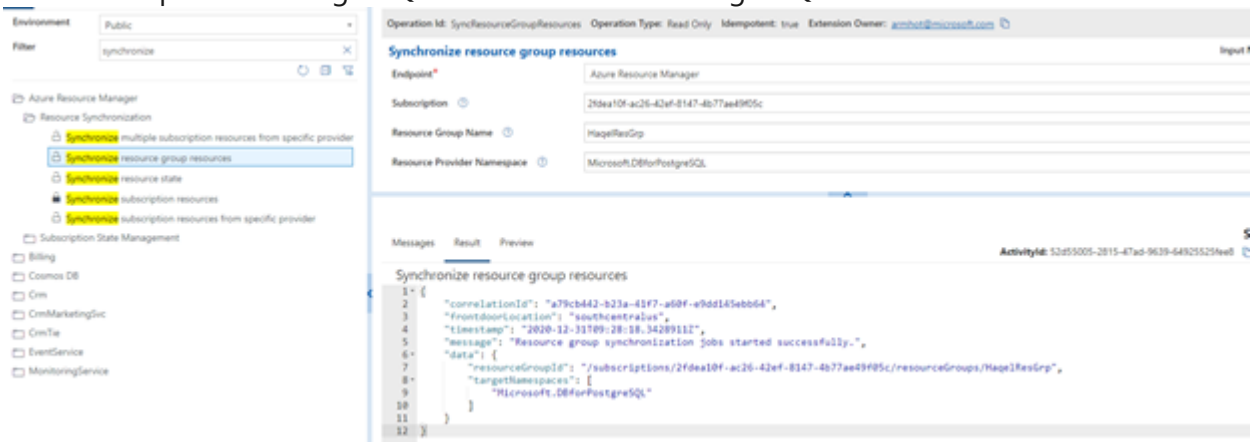
Note: Make sure to select the proper environment endpoint at the top-left corner. Set "Public" for regions like West Europe or Central US; set e.g., "Mooncake" for the China.

3. Azure resource Manager --> Resource Synchronization --> Synchronize resource group resources:



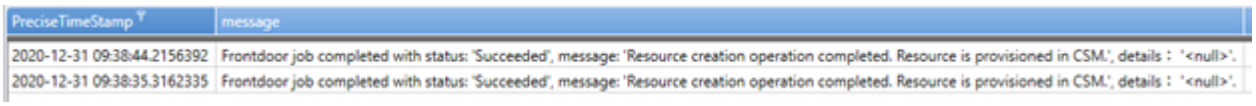
4. Enter the subscription, resource group, the namespace for the target resource, and click *Run*:

   - Namespace for MySQL is Microsoft.DBforMySQL
   - Namespace for MariaDB is Microsoft.DBforMariaDB
   - Namespace for PostgreSQL is Microsoft.DBforPostgreSQL



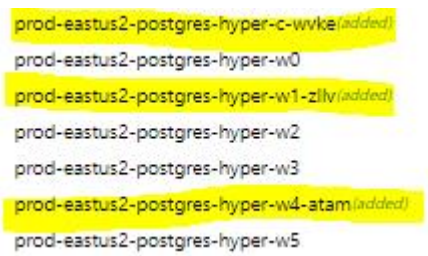
The sync will take a few minutes. Please make note of the correlationId as we can monitor it through Kusto query in [ARMProd cluster]

```
JobTraces
| where TIMESTAMP > ago(3h)
| where correlationId == "{}" // get from Jarvis shown in above screenshot
| where jobId contains "ResourceConsistencyJob"
| where message contains "Resource creation operation completed. Resource is provisioned in CSM."
| project PreciseTimeStamp , message
| sort by PreciseTimeStamp desc
```



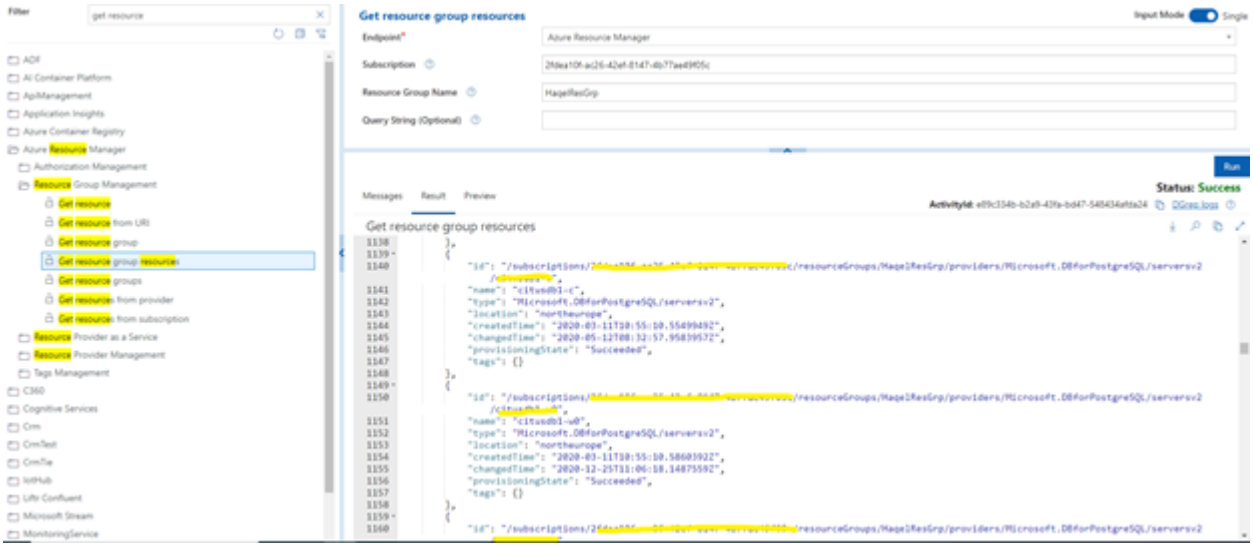
PreciseTimeStamp	message
2020-12-31 09:38:44.2156392	Frontdoor job completed with status: 'Succeeded', message: 'Resource creation operation completed. Resource is provisioned in CSM.', details : '<null>'.
2020-12-31 09:38:35.3162335	Frontdoor job completed with status: 'Succeeded', message: 'Resource creation operation completed. Resource is provisioned in CSM.', details : '<null>'.

5. Keep monitoring till the operation completed, once it is completed you should be able to see the resource in the ASC , with added note next to the resource:<

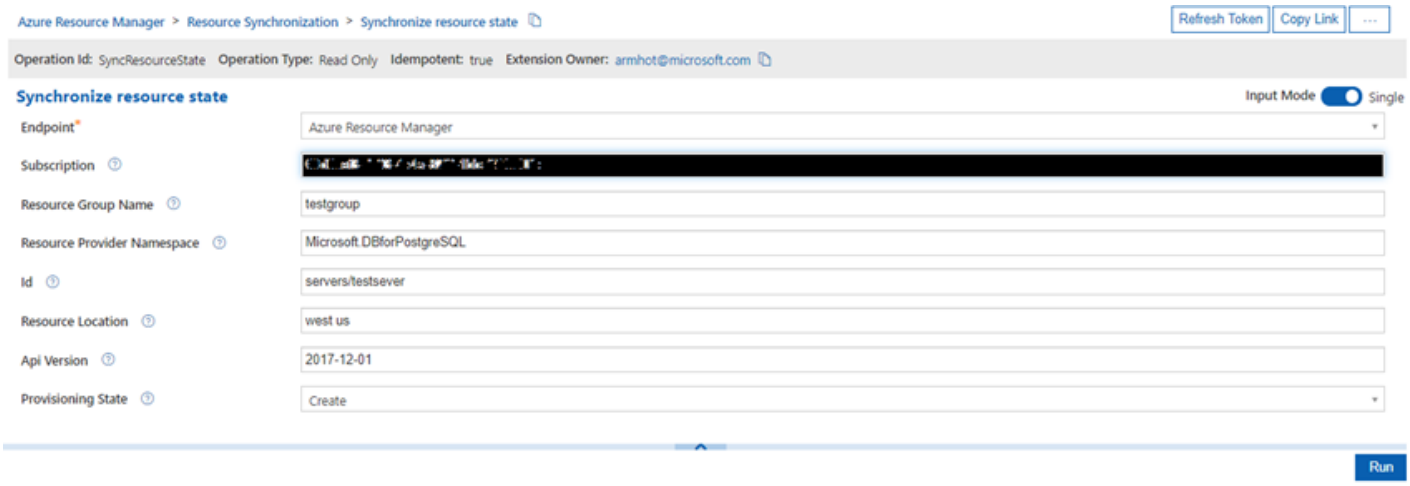


You can also check it from Jarvis: Azure Resource Manager --> Resource Group Management --> get resource group resources.

Fill the subscription ID and the resource group:



6. If resource(s) do not show up in the output, you can also try synchronize them specifically by using the "Synchronize Resource State" operation: Azure Resource Manager > Resource Synchronization > Synchronize resource state



Important: in Id above

- Use "servers/{servername}" for Single Server
- Use "flexibleServers/{servername}" for Flexible Server

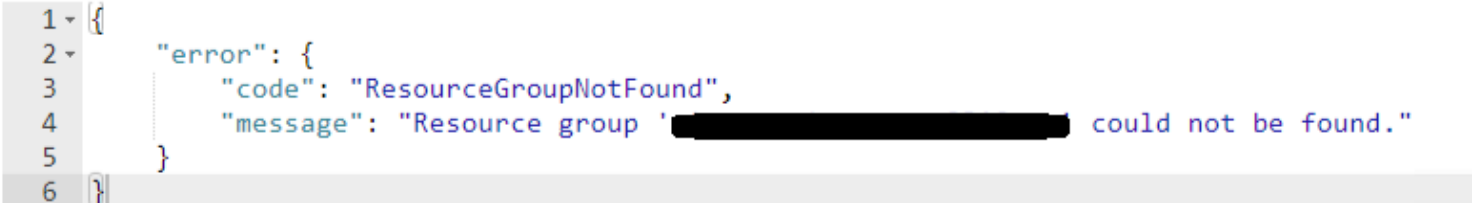
Known issues

Resource group 'xxxx' could not be found

If the sync failed with error "Resource group 'xxxx' could not be found" (as shown below) ask the Cx to recreate the resource group and then they the sync again

Messages Result Preview

Synchronize resource group resources



```
{
    "error": {
        "code": "ResourceGroupNotFound",
        "message": "Resource group '[redacted] could not be found."
    }
}
```

If you have any doubts when following this page, please reach out to *haaqel* for clarification and wiki/TSG improvement.