Recover a dropped resource

Last updated by | Pedro Acevedo | Apr 26, 2022 at 9:21 AM PDT

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

PostgreSQL Recover a dropped resource

Scenario:

If customer contacted you after he accidentally dropped a server asking us to help with recovering the server.

Set expectations:

Let customer know that server deletion is irreversible operation.
However, we will do our best efforts to help the customer get his server back.

For **CMK enabled** Servers -Cx won't be able to restore by themselves. Pleae work with CX to get the below 6 steps comple requested information.[only for CMK Enabled server, other servers customer can do it]

Request CX to create a new server with identity and assign newly created identity access to the same AKV that is required to be recovered

```
1. Please create a Postgres server temporarily from the portal or CLI(Get started with Azure CLI | Microsoft Docs) on the same subscription and resource_group
az postgres  server create --name <server name> --resource-group <resoure_group> --location <locations> --storage-size <size>  -u <user>-p <pwd> --backup-rete

2. If you are creating the server using the above command, step 2 is not required. Once the server is created, please use the below command to create a System
az postgres  server update –name <server name> -g <resoure_group> --assign-identity

3. Get/Verify the identity got created by looking at the identity response for principle ID
az postgres server show --name <server name> --resource-group <resoure_group>
"identity": {
    "principalId": "<Guid>",
    "tenantId": "<Guid>",
    "type": "SystemAssigned"
  },

4. Grant the server identity access to key vault (Assign an Azure Key Vault access policy (CLI) | Microsoft Docs)
az keyvault set-policy --name <name of the azure keyvault> -g <resoure_group> --key-permissions get unwrapKey wrapKey --object-id <principl id of the server f
You can do this from the portal as well - Assign an Azure Key Vault access policy (Portal) | Microsoft Docs)

5. Ensure new identity has access to the same key vault as the original server
6. Have CX  share the newly created server name with the engineering team
```

Customers can do it themselves

Using REST-API call customers can run CREATE command to restore their dropped server
https://docs.microsoft.com/en-us/rest/api/postgresql/singleserver/servers/create

Servers - Create

Service: MySQL

API Version: 2017-12-01

Creates a new server or updates an existing server. The update action will overwrite the existing server.

HTTP Copy Try It

```
PUT https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Micro
```

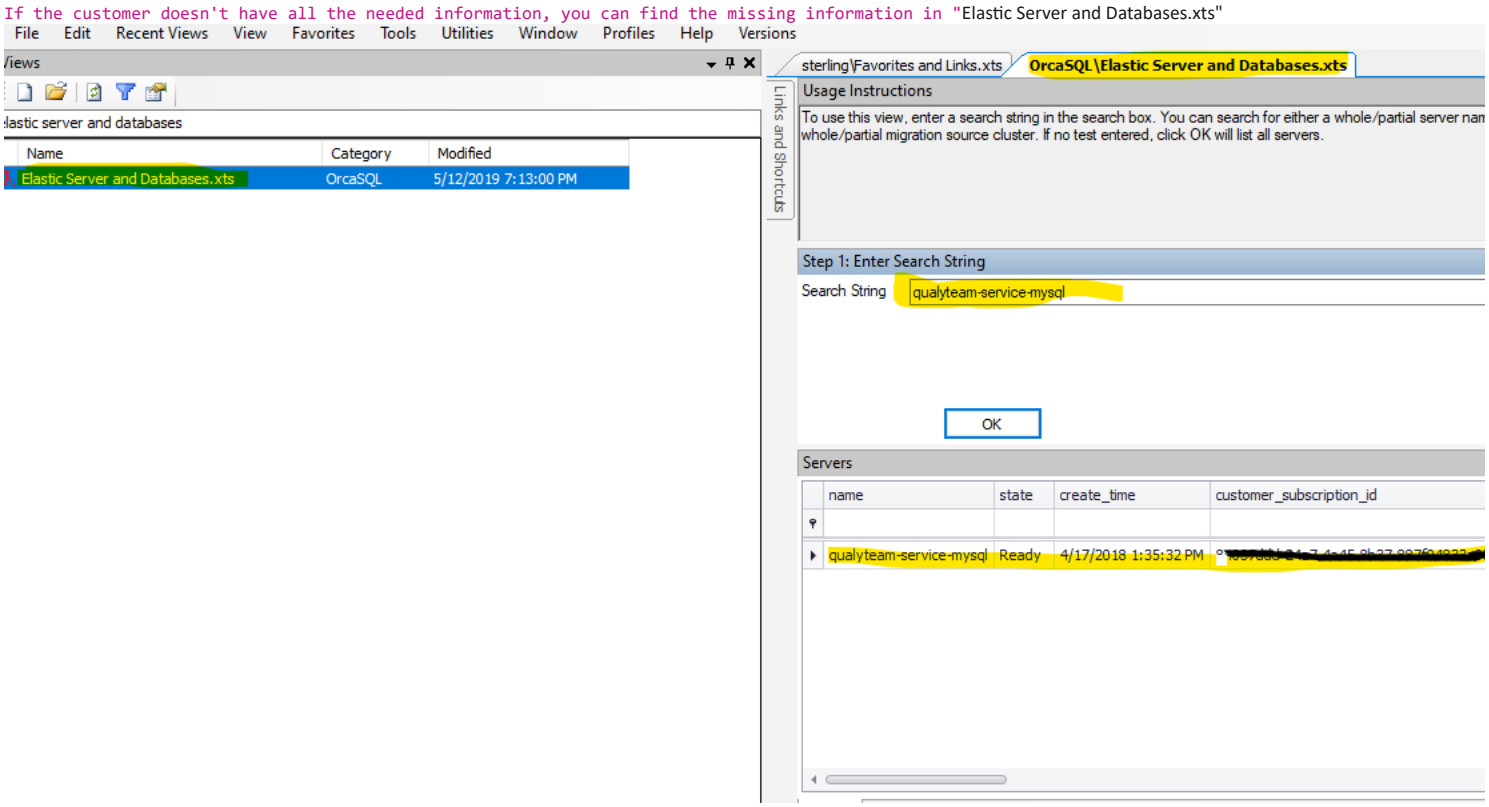
Data to be collect about the Dropped Server:

- Server Name
- Region where the server was hosted
- Subscription ID
- Resource Group where the server was located (make sure the RG is in place, if it was deleted as well, ask customer to create RG with the same name before the r
- Dropped date

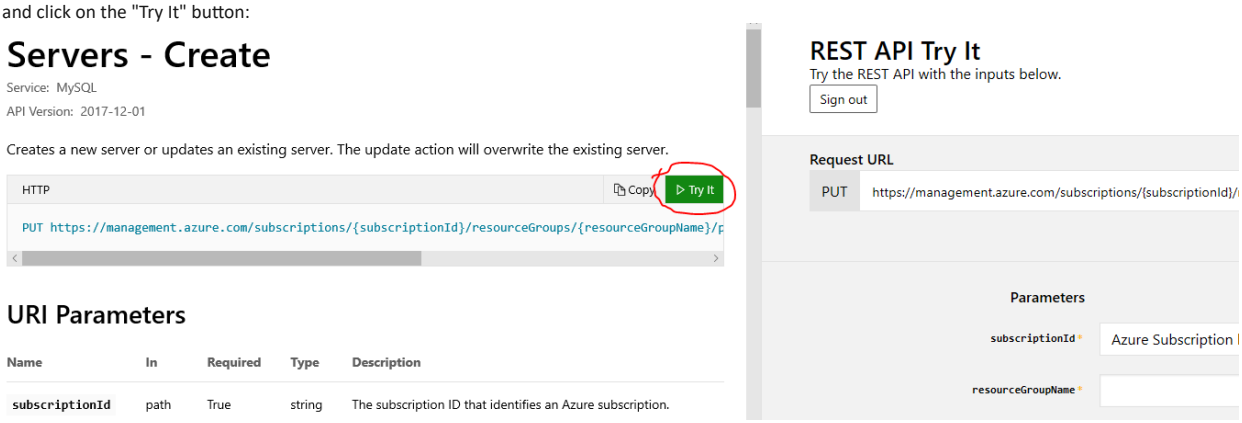
Run the following Kusto Query to get the Dropped date

```
//find the dropped date of a server
 MonAnalyticsElasticServersSnapshot
| where name == "server_name"
| summarize arg_max(TIMESTAMP, *) by elastic_server_id
| project name, elastic_server_id, ['state'],dropped_date
```

If the customer doesn't have all the needed information, you can find the missing information in "Elastic Server and Databases.xts"

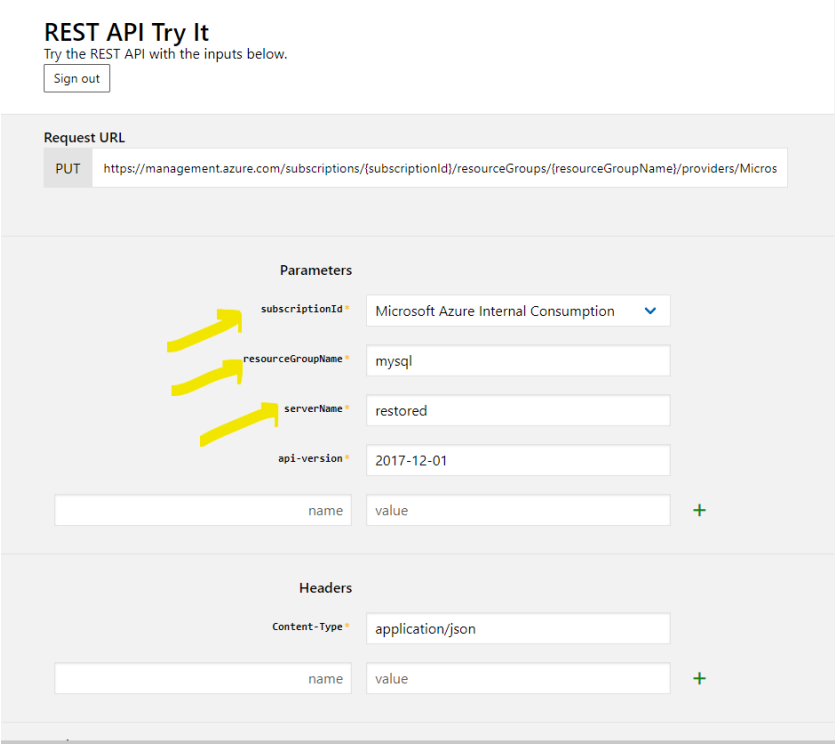


For example, go to: https://docs.microsoft.com/en-us/rest/api/postgresql/singleserver/servers/create
and click on the "Try It" button:



REST API CREATE

Once you collect all the needed information the customer should run the REST API create command as the following example



Regarding the Body
part of the REST API create command use the following as an example

```
{
 "location": "<Dropped Server Location>",
 "properties": {
  "restorePointInTime": "<Time less than dropped time>",

  "createMode": "PointInTimeRestore",
  "sourceServerId": "/subscriptions/<Dropped Server Subscription ID>/resourceGroups/<Dropped Server Resource Group Name>/providers/Microsoft.DBforPostgreSQL/servers/<Dropped
 }
}
```

Note: dropped server and new server can have same name
Example:

```
{
 "location": "SouthCentralUS",
 "properties": {
  "restorePointInTime": "2019-07-19T17:11:58Z",

  "createMode": "PointInTimeRestore",
  "sourceServerId": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/mysql/providers/Microsoft.DBforPostgreSQL/servers/restored"
 }
}
```

Kusto query to find Tombstoned server details

```
// Use this query if server is tombstoned
// tombstoned_to_dropdate -> After this time all the data will be gone. We can't restore the server
// min_restore_point ->Minimum Point In time we can restore to
// max_restore_point -> Max Point In time we can restore the server to
MonAnalyticsElasticServersSnapshot
| where name == "<Server Name>" // dropped server name
| summarize arg_max(TIMESTAMP, *) by elastic_server_id
| extend IsPFS = iif(fsm_extension_data contains '<PropertyName>IsPFS</PropertyName><Value>False</Value>', 'false', 'true')
| project name,elastic_server_id, ['state'], dropped_date, tombstoned_to_dropdate = datetime_add('day', 7 ,dropped_date),
min_restore_point = max_of(datetime_add('day', -backup_retention_days , now()),  create_time), max_restore_point = datetime_add('second', -1, dropped_da
resource_group, customer_subscription_id, elastic_server_type, AppName=physical_instance_name, elastic_server_edition, IsPFS
```

Global Kusto query to find all Tombstoned servers

First try all regions in public cloud.

```
let GlobalMonAnalyticsElasticServersSnapshot = union
  cluster('sqlazureau2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot        // australia
, cluster('sqlazurebr2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot        // brazil
, cluster('sqlazureca2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot        // canada
, cluster('sqlazurecus2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot       // central us
, cluster('sqlazureeas2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot       // east asia
, cluster('sqlazureeus12.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot      // east us1
```

```
    , cluster('sqlazureeus22.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot      // east us2
    , cluster('sqlazurefra.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot        // france
    , cluster('sqlazureince2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot      // india central
    , cluster('sqlazureinso2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot      // india south
    , cluster('sqlazureinwe2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot      // india west
    , cluster('sqlazureja2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot        // japan
    , cluster('sqlazurekor.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot        // korea
    , cluster('sqlazurencus3.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot      // north cenral us
    , cluster('sqlazureneu2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot       // north europe
    , cluster('sqlazurescus2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot      // south central us
    , cluster('sqlazureseas2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot      // south east asia
    , cluster('sqlazuresouthafrica.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot// south africa
    , cluster('sqlazureuk2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot        // uk
    , cluster('sqlazrwcus.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot         // west central us
    , cluster('sqlazureweu2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot       // west europe
    , cluster('sqlazurewus1.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot       // west us1
    , cluster('sqlazurewus2.kustomfa.windows.net').database('sqlazure1').MonAnalyticsElasticServersSnapshot       // west us2
    ;
    GlobalMonAnalyticsElasticServersSnapshot
    | where customer_subscription_id == toupper("<Subscription Id>") and ['state'] == 'Tombstoned'
    | summarize arg_max(TIMESTAMP, dropped_date, resource_group, name, ClusterName) by elastic_server_id
    | project DroppedDate = dropped_date, ResourceGroup = resource_group, LogicalServerName = name, ClusterName
```

Then try Mooncake/BlackForest/FairFax clouds separately.

```
    MonAnalyticsElasticServersSnapshot
    | where customer_subscription_id == toupper("<Subscription Id>") and ['state'] == 'Tombstoned'
    | summarize arg_max(TIMESTAMP, dropped_date, resource_group, name, ClusterName) by elastic_server_id
    | project DroppedDate = dropped_date, ResourceGroup = resource_group, LogicalServerName = name, ClusterName
```

How to find PointInTimeUTC -- use the last good backup time to restore the server PITR.

PointInTimeUtc = Min (`max_restore_point, max_point_in_time`)

`max_restore_point = you get his from 1st Query` [Kusto query to find Tombstoned server details](#) `step`

`max_point_in_time = you get this from below queries`

24HR PointInTimeUtc format <YYYY-MM-DDTHH:MM:SSZ>

PG Basic (`elastic_server_type =` PostgreSQL.Server.PAL`, elastic_server_edition = Basic`):

```
MonRdmsPgSqlSandbox
 | where LogicalServerName == "<Source Name>"
 | where text contains "archived transaction log file"
 | summarize arg_max(originalEventTimestamp, *) by LogicalServerName, AppName
 | project Last_successful_Backup=originalEventTimestamp, max_point_in_time =datetime_add('minute', -5, originalEventTimestamp), LogicalServerName, AppN
ResourceGroup, text
```

PG PFS (`elastic_server_type =` PostgreSQL.Server.PAL`, Is PFS = true`):

```
MonRdmsPgSqlSandbox
 | where LogicalServerName == "<Source Name>"
 | where text contains "[info] xlogcopy.archive: archived"
 | summarize arg_max(originalEventTimestamp, *) by LogicalServerName, AppName
 | project Last_successful_Backup=originalEventTimestamp, max_point_in_time =datetime_add('minute', -5, originalEventTimestamp), LogicalServerName, AppN
ResourceGroup, text
```

SBS Server (PG/MySQL/MariaDB) (`Is PFS = false and elastic_server_edition in (GeneralPurpose, MemoryOptimized)`):

```
AlrBackup
| where LogicalServerName == "<Source Name>"
 | where event_type == "BACKUP_METADATA_DETAILS"
 | summarize arg_max(originalEventTimestamp, *) by LogicalServerName, AppName
 | project originalEventTimestamp , LogicalServerName, AppName, backup_type , backup_start_date, backup_end_date, max_point_in_time= backup_end_date
```

`For all other Server types, max_point_in_time = max_restore_point`

`If above queries are not returning any value, max_point_in_time = max_restore_point`

FAQ:

- What if I don't know the cluster in which the server is located?

Run [Kusto query to find Tombstoned server details](#) Kusto query in "fanout query cross all clusters.xts", XTS view. It will output the cluster name

- **Server does not show up in the Azure portal after restore.**

Sometimes after the server restored and visible in XTS, is might not be visible for the customer on the Azure portal although its already up and running (and available f
In case this happen: please follow the instruction to sync ARM cache.
https://dev.azure.com/Supportability/AzureDBPostgreSQL/_wiki/wikis/AzureDBPostgreSQL/436366/Sync-or-Rehydrate-ARM-Cache

What if the restore failed??

Collect request ID from customer operation.

```
MonManagement
| where originalEventTimestamp > ago(12h)
| where request_id =~ "c860a3d0-31b3-4df6-80f3-d96e8dfc6959"
```

From the above table there is a column called operation_parameters, it has all the details

Check the Restore Progress?

run the below query after changing the server name

```
let PointInTimeRestoreRequests =
MonElasticServerRestoreRequests
| where source_elastic_server_name =~ 'spl-postrgre-server'
| where originalEventTimestamp > ago(7d) and operation_detail == 'Restore';
let PointInTimeRestoreRequestStartEvent =
PointInTimeRestoreRequests
| where state == 'CreatingServerForRestore';
let PointInTimeRestoreRequestEndEvent =
PointInTimeRestoreRequests
| where state == 'VerifyElasticServerReady';
PointInTimeRestoreRequestStartEvent
| join kind=leftouter PointInTimeRestoreRequestEndEvent on target_elastic_server_id
| extend State = iff(isempty(event_type1), 'InProgress', event_type1),
Duration = iff(isempty(originalEventTimestamp1), now(), originalEventTimestamp1)-originalEventTimestamp
| project StartTime = originalEventTimestamp, EndTime = originalEventTimestamp1, Duration,
TargetServerName = target_elastic_server_name, State, PointInTime = point_in_time
|sort by StartTime asc
```

StartTime	EndTime	Duration	TargetServerName	State	PointInTime
2020-04-03 14:24:02.8512085		03:28:08.2095070	spl-postrgre-server	InProgress	4/3/2020 9:56:00 AM

How good have you found this content?