Error 45181 ARM template deployment failure - resource does not exist

Last updated by | Holger Linke | Feb 28, 2023 at 1:56 AM PST

Contents

Unable to deploy resources using ARM - resource does not exist

Issue

The customer is creating and updating SQL resources through ARM templates. The ARM deployment is failing either persistently or intermittently with an error 45181:

> Error 45181
> Resource with the name '<resource name>' does not exist. To continue, specify a valid resource name

The output that is returned to the customer from the template execution might have these details:

```
{
"code": "DeploymentFailed",
"message": "At least one resource deployment operation failed. Please list deployment operations for details.
"details": [
{
"code": "40647",
"message": "Subscription '<subscription_id>' does not have the server 'servername'."
},
{
"code": "45181",
"message": "Resource with the name 'servername' does not exist. To continue, specify a valid resource name."
}
]
}
```

The resource name usually is the name of the SQL server, and this appears to be wrong because the SQL server had been created or updated shortly before, for example by applying a configuration change or creating a database.

Investigation

Get the SQL server name and the time of the error from the customer. Then check `MonManagement` in Kusto for the deployment history:

```
let srv = 'servername';
let startTime = datetime(2023-02-08 01:20:00Z);
let endTime = datetime(2023-02-08 02:00:00Z);
let timeRange = ago(1d);
MonManagement
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| where logical_server_name =~ srv
| where event !startswith "management_workflow_query"
| project originalEventTimestamp, subscription_id, logical_server_name, logical_database_name, event, request_
| project originalEventTimestamp, event, request_id, elapsed_time, rule_name, error_code, level, exception_typ
| order by originalEventTimestamp asc

Sample output:

originalEventTimestamp       event                                                         request_id
---------------------------- ------------------------------------------------------------ ---------------------
(...)
2023-02-08 01:23:33.2620807  management_workflow_firewall_rule_operation_start             D3A99C87-567C-47FF-97
2023-02-08 01:23:33.3652210  management_workflow_firewall_rule_operation_complete          D3A99C87-567C-47FF-97
2023-02-08 01:23:33.9680357  management_workflow_firewall_rule_operation_start             60A2A132-C430-4D39-A2
2023-02-08 01:23:34.0843391  management_workflow_firewall_rule_operation_complete          60A2A132-C430-4D39-A2
(...)
2023-02-08 01:26:30.2204324  management_workflow_create_logical_database_async_start       11929F31-16A1-434F-AE
2023-02-08 01:26:30.2419879  management_workflow_create_logical_database_async_complete    11929F31-16A1-434F-AE
2023-02-08 01:27:43.2053876  management_operation_create_logical_database_complete          11929F31-16A1-434F-AE
(...)
2023-02-08 01:30:03.1460025  management_workflow_drop_logical_database_async_start         F30E2B64-4FEC-49EE-B0
2023-02-08 01:30:03.1460101  management_workflow_drop_logical_database_start               F30E2B64-4FEC-49EE-B0
2023-02-08 01:30:03.1608045  management_workflow_drop_logical_database_async_complete      F30E2B64-4FEC-49EE-B0
2023-02-08 01:30:04.5074806  management_operation_drop_logical_database_complete           F30E2B64-4FEC-49EE-B0
2023-02-08 01:31:04.3472900  management_workflow_drop_logical_server_async_start           8836959E-C18D-43C1-BA
2023-02-08 01:31:04.3566185  management_workflow_drop_logical_server_async_complete        8836959E-C18D-43C1-BA
(...)
2023-02-08 01:31:07.4072062  management_workflow_firewall_rule_operation_start             267AD365-4B00-46FF-8A
2023-02-08 01:31:07.6074934  management_workflow_firewall_rule_operation_start             9ABF8619-F931-482B-8C
2023-02-08 01:31:07.8403687  management_workflow_firewall_rule_operation_start             0D04647B-C7DA-4CDC-BA
2023-02-08 01:31:07.8440817  management_workflow_firewall_rule_operation_failure           9ABF8619-F931-482B-8C
2023-02-08 01:31:07.8464982  management_workflow_firewall_rule_operation_failure           0D04647B-C7DA-4CDC-BA
2023-02-08 01:31:09.1763301  management_operation_drop_logical_server_complete             8836959E-C18D-43C1-BA
2023-02-08 01:31:17.5836945  management_workflow_firewall_rule_operation_failure           267AD365-4B00-46FF-8A
```

You can also check `MonManagementOperations` in Kusto for further details. Filter the SQL server name in `operation_parameters` using the brackets like `>servername<`. You can also search for specific events, operations, or errors as indicated in the commented lines:

```
let startTime = datetime(2023-02-08 01:00:00Z);
let endTime = datetime(2023-02-08 02:00:00Z);
let timeRange = ago(30d);
MonManagementOperations
| where TIMESTAMP >= startTime
| where TIMESTAMP <= endTime
//| where  TIMESTAMP >= timeRange
| where operation_parameters contains '>servername<'
//| where event == 'management_operation_failure'
//| where error_code == 45181
//| where operation_type == "UpsertFirewallRule"
| extend d=parse_xml(operation_parameters)
| extend ServerName=tostring(d.InputParameters.ServerName)
| extend RuleName=tostring(d.InputParameters.RuleName)
| limit 1000
| project originalEventTimestamp, request_id, event, operation_type, ServerName, RuleName, error_code, error_s
| order by originalEventTimestamp asc


Sample output:

originalEventTimestamp       request_id                             event                          operation_typ
---------------------------- -------------------------------------- ------------------------------ -------------
2023-02-08 01:28:51.4653221  5838BA73-CBF5-4542-A529-456B15538385  management_operation_start    UpsertLogical
2023-02-08 01:28:52.4480190  5838BA73-CBF5-4542-A529-456B15538385  management_operation_success  UpsertLogical
2023-02-08 01:29:01.4798104  AECAE590-490E-4C94-AFA2-3EDB938F1141  management_operation_start    UpdateLogical
2023-02-08 01:29:02.6618783  AECAE590-490E-4C94-AFA2-3EDB938F1141  management_operation_success  UpdateLogical
2023-02-08 01:30:03.1524127  F30E2B64-4FEC-49EE-B0EA-271D17CF130A  management_operation_start    DropLogicalDa
2023-02-08 01:30:04.5073094  F30E2B64-4FEC-49EE-B0EA-271D17CF130A  management_operation_success  DropLogicalDa
2023-02-08 01:31:04.3484641  8836959E-C18D-43C1-BAEE-3C203E1950C5  management_operation_start    DropLogicalSe
2023-02-08 01:31:07.3269354  B4E640E7-1858-4F86-B331-82F444C6CA20  management_operation_start    UpsertFirewal
2023-02-08 01:31:07.4094722  267AD365-4B00-46FF-8AD6-83F0B0D18808  management_operation_start    UpsertFirewal
2023-02-08 01:31:07.4109505  7BB8A436-9F9C-40DD-A29C-371B8AFAA99D  management_operation_start    UpsertFirewal
2023-02-08 01:31:07.6100384  9ABF8619-F931-482B-8C16-970BF280B0F5  management_operation_start    UpsertFirewal
2023-02-08 01:31:07.7373977  9ABF8619-F931-482B-8C16-970BF280B0F5  management_operation_failure  UpsertFirewal
2023-02-08 01:31:09.1761546  8836959E-C18D-43C1-BAEE-3C203E1950C5  management_operation_success  DropLogicalSe
2023-02-08 01:31:17.5434357  267AD365-4B00-46FF-8AD6-83F0B0D18808  management_operation_failure  UpsertFirewal
2023-02-08 01:31:17.5543485  7BB8A436-9F9C-40DD-A29C-371B8AFAA99D  management_operation_failure  UpsertFirewal
2023-02-08 01:31:18.7192624  B4E640E7-1858-4F86-B331-82F444C6CA20  management_operation_failure  UpsertFirewal
```

You may take one of the `request_id` values and filter all operations that are associated with it. This can be helpful if the issue is within the operation itself, or to see if the operation stopped at a specific step between `old_state` and `new_state`. It won't help you though if the cause is outside of this operation.

```
MonManagement
| where request_id in ("9ABF8619-F931-482B-8C16-970BF280B0F5")
| project originalEventTimestamp, operation_type, event, elapsed_time, old_state, new_state, operation_result,
| order by originalEventTimestamp asc

Sample output:

originalEventTimestamp       operation_type      event                                                elapsed_ti
---------------------------- ------------------- ---------------------------------------------------- ----------
2023-02-08 01:31:07.6074934                      management_workflow_firewall_rule_operation_start
2023-02-08 01:31:07.6075331                      fsm_starting_request
2023-02-08 01:31:07.6099285                      fsm_creating_state_machine
2023-02-08 01:31:07.6100384  UpsertFirewallRule  management_operation_start
2023-02-08 01:31:07.6659540                      fsm_executing_action
2023-02-08 01:31:07.6669782                      fsm_changed_state
2023-02-08 01:31:07.6673911                      fsm_executed_action
2023-02-08 01:31:07.7052072                      fsm_executing_action
2023-02-08 01:31:07.7060910                      fsm_changed_state
2023-02-08 01:31:07.7065509                      fsm_executed_action
2023-02-08 01:31:07.7362561                      fsm_executing_action
2023-02-08 01:31:07.7373977  UpsertFirewallRule  management_operation_failure                           00:00:00.
2023-02-08 01:31:07.7403560                      fsm_changed_state
2023-02-08 01:31:07.7407491                      fsm_executed_action
2023-02-08 01:31:07.8439223                      fsm_finished_request
2023-02-08 01:31:07.8440817                      management_workflow_firewall_rule_operation_failure 00:00:00.
```

Analysis

You can derive the cause of the issue by looking into the ARM deployment workflow that you have extracted in the "Investigation" section above. There are several possible scenarios that may lead to error 45181 "resource does not exist".

Scenario 1 - Issue within the customer ARM template

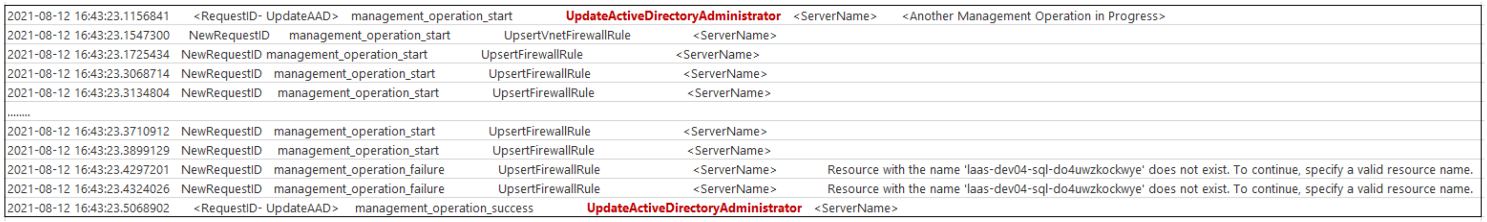
An example for this scenario is shown in the sample output from the "Investigation" section above:

- The `MonManagement` sequence starts with seemingly normal operations, like creating firewall rules and adding databases.
- Then the previously created database is dropped again (management_workflow_drop_logical_database_async_start).
- Shortly after, in `request_id` "8836959E-C18D-43C1-BAEE-3C203E1950C5", the server itself is dropped (management_workflow_drop_logical_server_async_start).
- While the Drop Server has not completed yet, several new firewall rules are created. These commands are then failing with "resource does not exist".
- While the firewall rule operations still continue, the Drop Server completes (management_operation_drop_logical_server_complete).

The same sequence is then confirmed in `MonManagementOperations`. If you filter on one of the failure `request_id` values though, it doesn't provide any further conclusions as the cause is outside of the failing operation.

Scenario 2 - Timing issue related to asynchronous resource deployment

In the following sample output, you can see that an operation for "UpdateActiveDirectoryAdministrator" has started shortly before a set of firewall rule operations. While the updateAADAdmin operation is still running and has not completed yet, the firewall rule operations are failing with "resource does not exist". The updateAADAdmin operation completes after the failures:



```
2021-08-12 16:43:23.1156841   <RequestID- UpdateAAD>  management_operation_start   UpdateActiveDirectoryAdministrator  <ServerName>   <Another Management Operation in Progress>
2021-08-12 16:43:23.1547300   NewRequestID   management_operation_start   UpsertVnetFirewallRule   <ServerName>
2021-08-12 16:43:23.1725434  NewRequestID management_operation_start   UpsertFirewallRule   <ServerName>
2021-08-12 16:43:23.3068714  NewRequestID   management_operation_start   UpsertFirewallRule   <ServerName>
2021-08-12 16:43:23.3134804  NewRequestID   management_operation_start   UpsertFirewallRule   <ServerName>
........
2021-08-12 16:43:23.3710912  NewRequestID   management_operation_start   UpsertFirewallRule   <ServerName>
2021-08-12 16:43:23.3899129  NewRequestID   management_operation_start   UpsertFirewallRule   <ServerName>
2021-08-12 16:43:23.4297201  NewRequestID   management_operation_failure   UpsertFirewallRule   <ServerName>   Resource with the name 'laas-dev04-sql-do4uwzkockwye' does not exist. To continue, specify a valid resource name.
2021-08-12 16:43:23.4324026  NewRequestID   management_operation_failure   UpsertFirewallRule   <ServerName>   Resource with the name 'laas-dev04-sql-do4uwzkockwye' does not exist. To continue, specify a valid resource name.
2021-08-12 16:43:23.5068902   <RequestID- UpdateAAD>   management_operation_success   UpdateActiveDirectoryAdministrator  <ServerName>
```

The cause of the failure is that the UpsertFirewall management operation requires the server state machine to be in "Ready" state before it executes successfully. Otherwise this operation will fail while looking for the target resource in the expected state.

The ARM deployment workflow is almost immediate and this seems to be a race condition between parallel management operations, here: the UpdateActiveDirectoryAdministrator vs. upsertfirewallrule. A reason for intermittent occurrences could be due to fine margins of timings involved (in milliseconds).

Mitigation

ARM template issues

This is covered by scenario 1 from above. The issue could be a genuine workflow design error by the customer; or it could arise from some conditional workflow or error handling within the template. Get the ARM template from the customer and match its steps to what you are seeing in `MonManagement` .

Timing issue - asynchronous resource deployment

This is covered by scenario 2 from above. Insert a dependency into the ARM template so that the deployment only continues after the previous blocking operation has completed.

This can be achieved with a "dependsOn" condition in the resource that needs to be delayed. See ARM template to create SQL DB server and database for an example. In scenario 2 from above, the UpsertFirewallRule would need to depend on the completion of the UpdateActiveDirectoryAdministrator.