Failing to deploy Postgres server using Terraform

Last updated by | Hamza Aqel | Feb 25, 2022 at 7:50 AM PST

This TSG is part of GT for any change please contact haaqel@microsoft.com

Contents

Issue

Trying to deploy a new Azure Postgres Single Server getting error message below consistently failing multiple times.

Error: [0m[0m[1mcreating PostgreSQL Server 'e-coretest-ccoe-frm0-tfe-alpha-postgres-cae-1' (Resource Group 'e-coretest-ccoe-frm0-tfe-alpha-data-rg-cae-1'): postgresql.ServersClient#Create: Failure sending request: StatusCode=0 -- Original Error: autorest/azure: Service returned an error. Status=nil Code='ServiceBusy' Message='Service is temporarily busy and the operation cannot be performed. Please try again later.'

Investigation/Analysis

Describes the steps/queries to use for confirming the issue Ask the customer these scoping questions:

1. Terrraform script and parameters
2. Share server_name and region trying to be deployed.
3. Json output of failed operation from activity log. If you have correlation id you can search it in the activity log and it will show up the operation failure and JSON.

- What is the order of creation of resources? E.g. db-server, replica, private endpoint, etc.
- Is there dependency established when deploying resources? E.g. replica depends on server creation, private endpoint depends on replica, etc.
- Has the customer already deployed same terraform script successfully in the past? If positive, try to find out what changed comparing previous good deployment versus current bad deployment.
- Has the customer tried deploying resources using ARM template to reproduce the error? If positive, please get ARM template of deployment to analyze failure.
- Is the deployment failing immediately after trying to create server or when trying to create other dependencies?

Run the following Kusto query in MonManagement to get the information on why request failed:

```
MonManagement
| where TIMESTAMP>= datetime(2021-07-07 13:40)
| where TIMESTAMP<= datetime(2021-07-07 13:45)
| where subscription_id=~ "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
| where request_id in("8E90AFAD-71FF-46AD-9324-77CE3BA981CF")
| where elastic_server_name == "e-coretest-ccoe-frm0-tfe-pr-postgres-cac-1 "
| project originalEventTimestamp, request_id, ['state'], error_code, error_message, operation_type, operation_
| order by request_id, originalEventTimestamp asc
```

If you see this behavior:

1. First operation request on server is being processed. Initial state server is busy.
2. Other incoming requests are triggered simultaneously after the first operation, trying to run same operation on the server which is currently busy and all three request failed.
3. Finally, the initial operation highlighted in yellow below competes.

originalEventTimestamp	request_id	state	error_code	error_message	operation_type
2021-07-07 20:36:38.4	AFE991E3-C0F2-46B0-BDF8-BBB47221A579	Succeeded	0		DropElasticServer
2021-07-07 20:42:58.0	8E40CED2-ED9B-428D-BF14-365AD0CB2B6B	Busy			UpsertElasticServer
2021-07-07 20:43:29.6	FF630BB2-6FFC-45DE-A0A9-B6CEC74B7976				UpsertElasticServer
2021-07-07 20:43:30.0	FF630BB2-6FFC-45DE-A0A9-B6CEC74B7976	Failed	45157	Server 'e-coretest-ccoe-frm0-tfe-pr-postgres-cac-1' is busy with another operation. Please try your operation later.	UpsertElasticServer
2021-07-07 20:43:30.8	78BDCBE4-7651-4591-824A-D2A1D7561727				UpsertElasticServer
2021-07-07 20:43:31.2	78BDCBE4-7651-4591-824A-D2A1D7561727	Failed	45157	Server 'e-coretest-ccoe-frm0-tfe-pr-postgres-cac-1' is busy with another operation. Please try your operation later.	UpsertElasticServer
2021-07-07 20:43:32.0	422D645E-1BFA-4418-81CE-F06187ECFC24				UpsertElasticServer
2021-07-07 20:43:32.5	422D645E-1BFA-4418-81CE-F06187ECFC24	Failed	45157	Server 'e-coretest-ccoe-frm0-tfe-pr-postgres-cac-1' is busy with another operation. Please try your operation later.	UpsertElasticServer
2021-07-07 20:44:21.9	8E40CED2-ED9B-428D-BF14-365AD0CB2B6B	Succeeded	0		UpsertElasticServer

The Customer was using Terraform with Jenkins pipeline where we notice timeouts calling the Rest API on each request:

Please notice that request is not getting response, there is a conflict (as server is already being created), and ServiceBusy is because initial request is being processed. The terraform deployment is not able to get a response back from MS Rest API on the initial request causing multiple retries of server creation operation.

1. [2021-07-12T18:46:30.278Z] 2021-07-12T18:46:30.145Z [DEBUG] plugin.terraform-provider-azurerm_v2.67.0_x5: Request to https://management.azure.com/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/rg/providers/Microsoft.DBForPostgreSQL/servers/e-coretest-ccoe-frm0-tfe-alpha-postgres-cae-1?api-version=2017-12-01 **completed with no response**: timestamp=2021-07-12T18:46:30.143Z
2. [2021-07-12T18:47:02.549Z] 2021-07-12T18:47:02.521Z [DEBUG] plugin.terraform-provider-azurerm_v2.67.0_x5: AzureRM Response for https://management.azure.com/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/rg/providers/Microsoft.DBForPostgreSQL/servers/e-coretest-ccoe-frm0-tfe-alpha-postgres-cae-1?api-version=2017-12-01: [2021-07-12T18:47:02.549Z] HTTP/1.1 409 **Conflict**
3. [2021-07-12T18:47:05.593Z] 2021/07/12 18:47:05 [DEBUG] module.postgres-sql.azurerm_postgresql_server.main: apply errored, but we're indicating that via the Error pointer rather than returning it: creating PostgreSQL Server "e-coretest-ccoe-frm0-tfe-alpha-postgres-cae-1" (Resource Group "rg"): postgresql.ServersClient#Create: Failure sending request: StatusCode=0 -- Original Error: autorest/azure: Service returned an error. Status=<nil> Code="ServiceBusy" **Message="Service is temporarily busy and the operation cannot be performed. Please try again later."**

Mitigation

After sharing investigation/analysis above, the customer reached out internal Network Team to review what was blocking MS Rest API responses and discover that his own firewall was the cause of this issue because was detected as security vulnerability.

Below is what customer's internal team advised on why it is marking it as threat:

Looking at the HTTP RFC, the 'Retry-After' field should only be used within 503 (Service Unavailable), 429 (Too Many Request) and 301 (Moved Permanently). Having a Retry-After field on type 202 (Accepted) not only should not be used, but it also opens up the vulnerability for this signature (2019-17555).

If a permanent fix is desired, the administrator of this server needs to validate why this field is being used on 202 messages.

From the firewall standpoint, this signature is triggering as expected.

As per Microsoft “they confirmed that the "Retry-After" in the response header of the PUT request is by design”

Repro Steps

To support your findings and avoid extensive troubleshooting of many factors such as client software version, complex Terraform deployments using Jenkins pipelines, Network, on-premise machine configurations and large logs analysis ask the customer to run a basic example terraform script deployment to create server in cloud shell. If deployment completes successfully then issue is on customer setup configuration side.

Create three files:

variables.tf.txt

provider.tf.txt

main.tf

```
resource "random_integer" "deployment" {
  min = 10000
  max = 99999
}

resource "azurerm_resource_group" "rg" {
  name     = var.resourceGroupName
  location = var.location
}

resource "azurerm_postgresql_server" "postgresqlServer" {
  name                = "$ {var.serverName}-$ {random_integer.deployment.result}"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  administrator_login          = "psqladminun"
  administrator_login_password = "H@Sh1CoR3!"
  sku_name   = "GP_Gen5_4"
  version    = "11"
  storage_mb = 640000
  backup_retention_days        = 7
  geo_redundant_backup_enabled = false
  auto_grow_enabled            = false
  public_network_access_enabled    = false
  ssl_enforcement_enabled          = true
  ssl_minimal_tls_version_enforced = "TLS1_2"
}

resource "azurerm_postgresql_database" "postgresqlDatabase" {
  count               = 1
  name                = "database-${count.index}"
  resource_group_name = azurerm_resource_group.rg.name
  server_name         = azurerm_postgresql_server.postgresqlServer.name
  charset             = "UTF8"
  collation           = "English_United States.1252"
}
```

Run the following commands in cloud shell:

az login

az account set --subscription="subscription-id"

terraform init

OR

terraform init -upgrade

terraform plan -out pg.tfplan

terraform apply pg.tfplan

terraform plan -destroy -out pg.destroy.tfplan

terraform apply "pg.destroy.tfplan"

Root Cause Classification

This TSG should be coded to the following root cause:

Support area path: Azure/Azure Database for PostgreSQL single server/Create, Update and Drop Resources/Databases

Root cause classification: /Root Cause: PostgreSQL Single Server/Portal, Client Tools and APIs/Rest API

How good have you found this content?