Read Replicas

Last updated by | Hamza Aqel | Jan 9, 2023 at 9:00 AM PST

Contents

This is part of GT and under construction, for feedback or update, please refer to haaqerl@microsoft.com

The read replica feature allows you to replicate data from an Azure Database for PostgreSQL server to a read-only replica. Replicas are updated asynchronously with the PostgreSQL engine native physical replication technology. Streaming replication by using replication slots is the default operation mode. When necessary, log shipping is used to catch up. You can replicate from the primary server to up to five replicas.

Replicas are new servers that you manage similar to regular Azure Database for PostgreSQL servers. For each read replica, you're billed for the provisioned compute in vCores and storage in GB/ month.

Azure Database for PostgreSQL flexible server read replica/Geo replica is in **preview** now.

ETA for GA is Q1CY23 (Jan-Mar).

The below queries will help as temporary situation till the ASC and this TSG completed.

Master Replica Server pair

Use the below queries to understand the replica servers that are paired with the primary if the customer did not share with you and you are not aware of the replica region:

1- To check the Local replicas:

```
let c1= MonOBPgSqlReplicationStats
        | where strlen(Application_name) > 0
        | summarize arg_max(TIMESTAMP, Application_name) by LogicalServerName
```

```
                | extend VMNAME= tostring(split(Application_name, "azure_asyncreplica_")[-1])
                | extend Master =  LogicalServerName, ReplicaTemp = Application_name
                | where ReplicaTemp == 'haqelm21' or Master == 'haqelm21'
                | project LatestTime = TIMESTAMP, Master, ReplicaTemp,VMNAME;
                let c2 = MonPgLogs
| where VirtualMachineName in (
 (MonOBPgSqlReplicationStats
                | where strlen(Application_name) > 0
                | summarize arg_max(TIMESTAMP, Application_name) by LogicalServerName
                | extend VMNAME= tostring(split(Application_name, "azure_asyncreplica_")[-1])
                | project VMNAME
                ))
                | extend Replica = LogicalServerName
                | project Replica,VirtualMachineName;
  c1
| join kind= leftouter (
    c2
    )
    on $left.VMNAME == $right.VirtualMachineName
  | distinct  LatestTime,Master,Replica
```

Check the geo replicas:

et the VM name(s):

```
MonOBPgSqlReplicationStats
| where LogicalServerName == "haqelm21"
| summarize arg_max(TIMESTAMP, Application_name) by LogicalServerName
| extend VMNAME= tostring(split(Application_name, "azure_asyncreplica_")[-1])
| distinct  VMNAME
```

VMNAME
eadbff284033

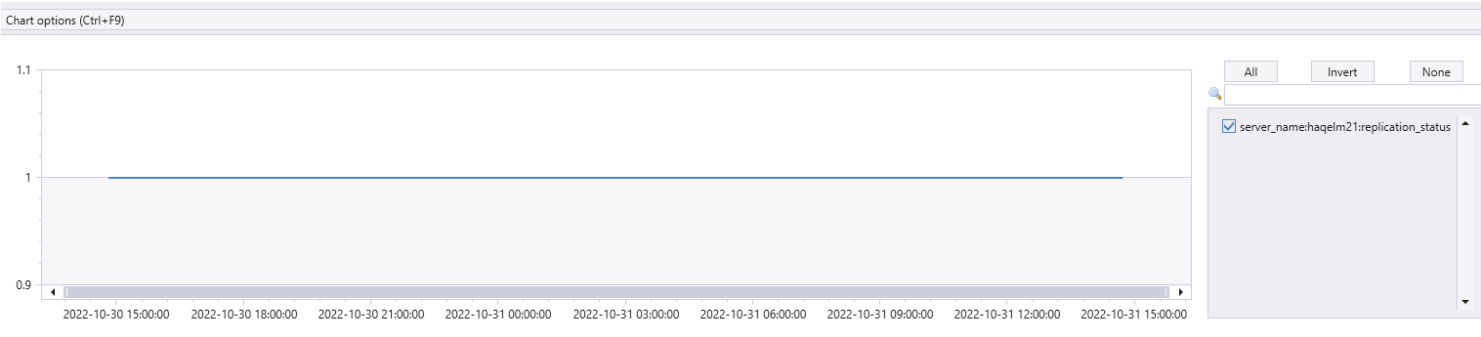
ɯn the below to execute it on all clusters:

```
let GlobalMonPgLogs = union
  cluster('sqlazureau2.kustomfa.windows.net').database('sqlazure1').MonPgLogs        // australia
, cluster('sqlazurebr2.kustomfa.windows.net').database('sqlazure1').MonPgLogs        // brazil
, cluster('sqlazureca2.kustomfa.windows.net').database('sqlazure1').MonPgLogs        // canada
, cluster('sqlazurecus2.kustomfa.windows.net').database('sqlazure1').MonPgLogs       // central us
, cluster('sqlazureeas2.kustomfa.windows.net').database('sqlazure1').MonPgLogs       // east asia
, cluster('sqlazureeus12.kustomfa.windows.net').database('sqlazure1').MonPgLogs      // east us1
, cluster('sqlazureeus22.kustomfa.windows.net').database('sqlazure1').MonPgLogs      // east us2
, cluster('sqlazurefra.kustomfa.windows.net').database('sqlazure1').MonPgLogs        // france
, cluster('sqlazureince2.kustomfa.windows.net').database('sqlazure1').MonPgLogs      // india central
, cluster('sqlazureinso2.kustomfa.windows.net').database('sqlazure1').MonPgLogs      // india south
, cluster('sqlazureja2.kustomfa.windows.net').database('sqlazure1').MonPgLogs        // japan
, cluster('sqlazurekor.kustomfa.windows.net').database('sqlazure1').MonPgLogs        // korea
, cluster('sqlazurencus3.kustomfa.windows.net').database('sqlazure1').MonPgLogs      // north cenral us
, cluster('sqlazureneu2.kustomfa.windows.net').database('sqlazure1').MonPgLogs       // north europe
, cluster('sqlazurescus2.kustomfa.windows.net').database('sqlazure1').MonPgLogs      // south central us
, cluster('sqlazureseas2.kustomfa.windows.net').database('sqlazure1').MonLogin      // south east asia
, cluster('sqlazuresouthafrica.kustomfa.windows.net').database('sqlazure1').MonPgLogs// south africa
, cluster('sqlazureuk2.kustomfa.windows.net').database('sqlazure1').MonPgLogs        // uk
, cluster('sqlazrwcus.kustomfa.windows.net').database('sqlazure1').MonPgLogs         // west central us
, cluster('sqlazureweu2.kustomfa.windows.net').database('sqlazure1').MonPgLogs       // west europe
, cluster('sqlazurewus1.kustomfa.windows.net').database('sqlazure1').MonPgLogs       // west us1
, cluster('sqlazurewus2.kustomfa.windows.net').database('sqlazure1').MonPgLogs       // west us2
, cluster('sqlazureusw3.westus.kusto.windows.net').database('sqlazure1').MonPgLogs       // west us3
;
GlobalMonPgLogs
| where VirtualMachineName in ("eadbff284033")
| distinct LogicalServerName,Region
```

LogicalServerName	Region
haqelm21rep	westeurope

Replication Status

```
let TimeCheckStart = datetime('2022-10-30 14:50:41');

        let TimeCheckEnd = datetime('2022-10-31 14:50:41');

        MonOBPgSqlReplicationStats

        | where PreciseTimeStamp >= TimeCheckStart and PreciseTimeStamp <= TimeCheckEnd and LogicalServerName =~ 'haqelm21'

| where Slot_name contains "asyncreplica"

        | extend result = case(Wal_sender_state =~ 'streaming', 1, 0)

        | summarize streaming_status = case(sum(result) >= 1, 1, 0) by bin(PreciseTimeStamp, 5m), server_name = LogicalServerName

        | project TIMESTAMP = PreciseTimeStamp, server_name, replication_status = streaming_status

        | order by TIMESTAMP asc

        | render timechart
```



To take an overall look:

```
MonOBPgSqlReplicationStats

| where LogicalServerName == "haqelm21"

| where Slot_name contains "asyncreplica"

| project  LogicalServerName, Slot_name, State, Slot_type, Wal_sender_state, PreciseTimeStamp, Application_name, Total_lag_in_bytes, Receiving_lag_in_bytes, Replaying_lag_in_bytes, Current_wal_lsn, Sent_lsn, Restart_lsn, Sending_lag_in_bytes, Write_lsn, Write_lag, Flush_lsn,Flush_lag, Replay_lsn, Replay_lag

| order by PreciseTimeStamp desc
```

LogicalServerName	Slot_name	State	Slot_type	Wal_sender_state	PreciseTimeStamp	Application_name
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 16:12:41.8704720	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 16:07:41.7988130	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 16:02:41.7250400	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 15:57:41.6600400	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 15:52:41.5882290	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 15:47:40.5513110	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 15:42:40.4681990	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 15:37:39.8127210	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 15:32:39.7532020	azure_asyncreplica_eadb
haqelm21	azure_asyncreplica_eadbff284033		physical	streaming	2022-10-31 15:27:39.6586780	azure_asyncreplica_eadb

Replication Lag

OBvmagentsidecarpgsql

| where TIMESTAMP >= ago(3d)

| where LogicalServerName == "haqelm21rep"

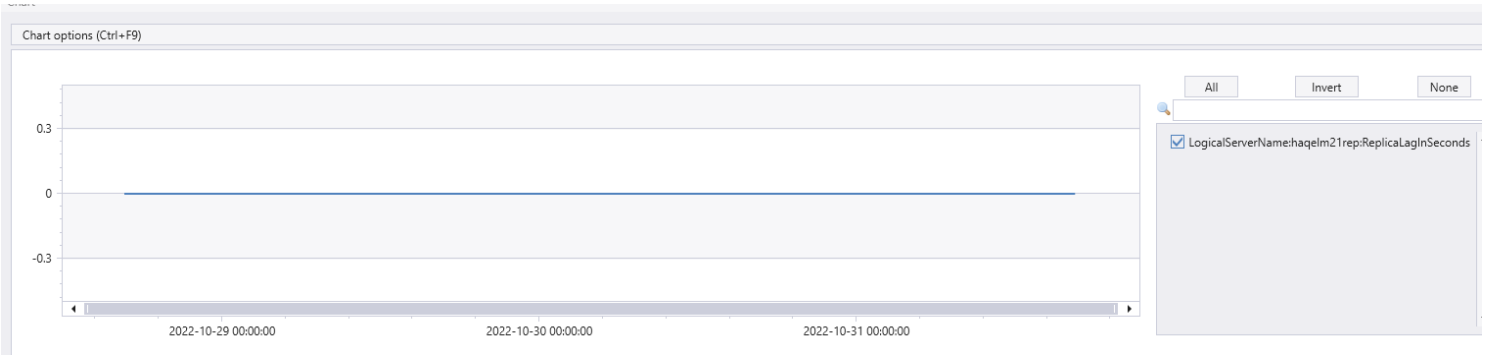
| where MessageString contains "SendShoeboxMetricValue: replica_log_delay_in_seconds = 0"

| extend replica_log_delay_in_seconds= tostring(split(MessageString, "replica_log_delay_in_seconds = ")[-1])

|extend ReplicaLagInSeconds = todecimal(substring(replica_log_delay_in_seconds, 0, 1))

| project TIMESTAMP,LogicalServerName,ReplicaLagInSeconds

| render timechart



Or you can use the previous query:

MonOBPgSqlReplicationStats

| where LogicalServerName == "haqelm21"

| where Slot_name contains "asyncreplica"

| project LogicalServerName, Slot_name, State, Slot_type, Wal_sender_state, PreciseTimeStamp, Application_name, Total_lag_in_bytes, Receiving_lag_in_bytes, Replaying_lag_in_bytes, Current_wal_lsn, Sent_lsn, Restart_lsn, Sending_lag_in_bytes, Write_lsn, Write_lag, Flush_lsn,Flush_lag, Replay_lsn, Replay_lag

| order by PreciseTimeStamp desc

..

From the customer side, he can run the below queries from the replica server:

SELECT CASE WHEN pg_catalog.pg_is_in_recovery()='t' THEN CASE WHEN pg_catalog.pg_last_wal_receive_lsn() =pg_catalog.pg_last_wal_replay_lsn() THEN 0 ELSE EXTRACT (EPOCH FROM now() - pg_catalog.pg_last_xact_replay_timestamp())::bigint END END AS replica_log_delay_in_seconds;

Or

SELECT EXTRACT (EPOCH FROM now() - pg_last_xact_replay_timestamp());

Lag metrics will be available in Azure portal after the M23 deployment, ETA first week of Nov.

Errors creating replica

If below was raised:

```
 "Can not found any vnet with name [REPLICA NAME] and resource id [RESOURCE ID] in resource group [RESOURCE GROUP NAME] with subscription [SUBSCRIPTION ID]"
```

Make sure:

- Provided correct vnet
- if customer using vnet peering between source and replica vnets then make sure NSG rules configured to allow replica to communicate with primary.
- VNET peering exists in case of geo replication scenario.

- Customer is using some other mechanism instead of vnet peering like hub and spoke configuration then NSG rules configured to allow replica to communicate with primary.

Note:

Please follow our documentation for more details about this feature,limitations,..etc