How to check the server long recovery and ETA to compete

Last updated by | Hamza Aqel | Mar 8, 2023 at 2:38 AM PST

Sometimes, a PostgreSQL could be unavailable for a long time after a restart or failover(The cause of restart could be referred at [CSSTSG-Orcas-Sterling-Restart] Why Sterling Server is restarted). In most cases, the duration should be related to that server was in a recovery mode. This TSG will illustrate

Contents

How to confirm if sever is in a recovery mode and taking long

In sandbox, run below query to get the keyword ***"redo starts at"*** or ***"redo done at "*** . The gap between these two message will be the time used for recovery. If the last message is ***"redo starts at"*** , it means the server is still in the process of recovery.

```
2022-04-29 01:35:38 UTC-626b40e7.2c-LOG: redo starts at 11A/82284500
2022-04-29 06:22:01 UTC-626b40e7.2c-LOG: redo done at 129/92D05F78
2022-04-29 06:22:02 UTC-626b40e7.2c-LOG: database startup complete in 17187 seconds, startup began 93390 secon
OrcasqlInstanceProvider::FabricChangeRole Database port open signaled.
OrcasqlInstanceProvider: FabricChangeRole (currentRole = ActiveSecondary, newRole = Primary) finished, duratio
OrcasqlInstanceProvider::FabricChangeRole unavailability duration 17220384 ms
SqlSocketDuplicateProvider::DB engine is ready.
2022-04-29 06:22:03 UTC-626b40dd.20-LOG: database system is ready to accept connections
```

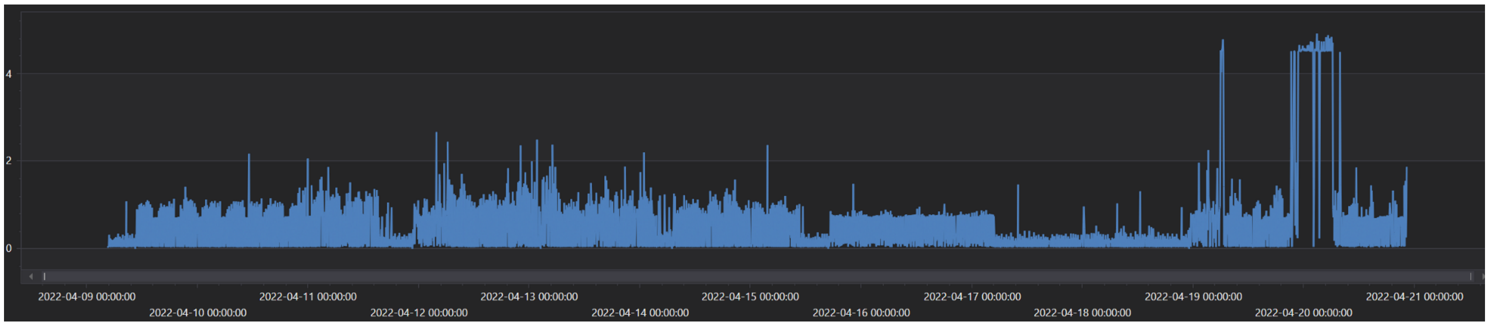
You can confirm via running below KQL.

```
cluster('sqlazureau2.kustomfa.windows.net').database('sqlazure1').
MonRdmsPgSqlSandbox
//| where originalEventTimestamp >= datetime(2021-11-11 01:15:58.7155259) and originalEventTimestamp <= dateti
| where LogicalServerName =~ '{servername}'
| where text startswith "Loaded package:  [PostgreSQL"or text startswith "SbsProvider::IsSbsXioStorage: Is SBS
    or text startswith "SbsProvider::IsSbsPfsStorage: Is SBS_PFS"
    or text contains "PostgresLauncher: DoCheckpointPG: "
    and text contains "Opening connection to PG for checkpoint"
    or text contains "PostgresLauncher: DoCheckpointPG: "
    and text contains "PG checkpoint in progress"
    or text contains "PostgresLauncher: DoCheckpointPG: "
    and text contains "Time taken in PG checkpoint is"
    or text contains "PostgresLauncher: Warning: DoCheckpointPG: "
    and text contains "PG checkpoint failed"
    or text contains "PostgresLauncher: DoCheckpointSBS: SBS checkpoint in progress"
    or text contains "PostgresLauncher: DoCheckpointSBS: Time taken in SBS checkpoint is"
    or text contains "PostgresLauncher: ShutdownWithSFIntegration: SIGINT has been sent"
    or text contains "PostgresLauncher: ShutdownWithSFIntegration: SIGQUIT has been sent"
    or text contains "Parallel redo is started for database 'SBS'"
    or text contains "Parallel redo is shutdown for database 'SBS'"
    or text contains "Recovery completed for database SBS"
    or text contains "LOG:  checkpoint complete"
    or text contains "LOG:  checkpoint starting: time"
    or text contains "LOG:  checkpoint starting: shutdown immediate"
    or text contains "LOG:  database system is shut down"
    or text contains "NOTICE:  database system is shut down"
    or text contains "LOG:  database system was not properly shut down"
    or text contains "LOG:  entering standby mode"
    or text contains "LOG:  redo starts at"
    or text contains "LOG:  consistent recovery state reached at"
    or text contains "LOG:  database system is ready to accept read only connections"
    or text contains "FATAL:  could not connect to the primary server"
    or text contains "LOG:  started streaming WAL from primary"
    or text contains "LOG:  trigger file found: " // PromoteTriggerFile.txt
    or text contains "LOG:  redo done at"
    or text contains "LOG:  checkpoint starting: end-of-recovery immediate"
    or text contains "LOG:  database startup complete in"
    or text contains "LOG:  database system is ready to accept connections"
    or text contains "SqlSocketDuplicateProvider::DB engine is ready"
    or text contains "Postgres startup parameters"
    or text contains "Listen to port"
    and text contains "at checkpoint"
    or tolower(text) contains "role"
    and tolower(text) contains "orcasql"
    and tolower(text) !contains "Wait on"
| order by originalEventTimestamp asc
| project originalEventTimestamp, NodeName, process_id, text
```

How to confirm if it is caused by workload increase or previous checkpoint failure

Use the keyword ***"Checkpoint completed with redo LSN"*** to the most recent successful checkpoint. You can also use below query to see the all past checkpoint time and time to see the most recent checkpoint time

```
cluster('sqlazureau2.kusto.windows.net').database('sqlazure1').
MonRdmsPgSqlSandbox
//| where originalEventTimestamp > ago(5d)
| where LogicalServerName == "{servername}"
| where text contains "checkpoint complete" and text contains "total="
| extend checkpoint_time_in_mins = todouble(replace(" s.*", "", replace("^.*total=", "", text)))/60
| project originalEventTimestamp, LogicalServerName, checkpoint_time_in_mins, text
| render timechart
```



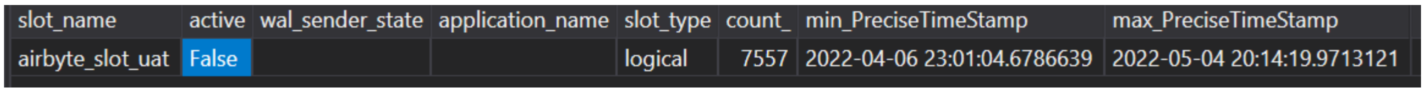
- If the checkpoint time is up-to-date, please run below KQL to check if Detecting workload increases or decreases and see if the long recovery is caused by heavy workload.

```
MonDmPgSqlXlogFileCount
| where LogicalServerName == "{Servername}}"
| extend index = indexof(pg_current_wal_lsn, '/', 0, -1, 1)
| extend offset = substring(pg_current_wal_lsn, index + 1)
| extend len = 8 - strlen(offset)
| extend walFileName = strcat('00000001', strrep('0', 8-index), substring(pg_current_wal_lsn, 0,index), '
| extend logicalXLogFile = strcat('0x',substring(walFileName, 8, 8))
| extend logicalXLogFileInt = toint(logicalXLogFile)
| extend xlogSegment = strcat('0x',substring(walFileName, 16, 8))
| extend xlogSegmentInt = toint(xlogSegment)
| extend walFileNumber = logicalXLogFileInt * 255 + xlogSegmentInt
| order by PreciseTimeStamp asc
| project  LogicalServerName, PreciseTimeStamp, pg_current_wal_lsn, walFileName, offset, logicalXLogFile,
| summarize maxWalFileNumberForHour = max(walFileNumber) by LogicalServerName, bin(PreciseTimeStamp, 1h)
| order  by PreciseTimeStamp asc
| serialize  PreciseTimeStamp, maxWalFileNumberForHour, prevMaxWalFileNumberForHour = prev(maxWalFileNumb
| extend FinalDistance = iff(Distance> 10000, 0, Distance)
| project  LogicalServerName, PreciseTimeStamp, FinalDistance
| render timechart
```

> Note that you can do FinalDistance * 16MB to get the size of the transaction logs produced per hour

- If the checkpoint time is sometimes ago, it should be caused by that checkpoint failed for some reasons. It is most likely there are inactive replications slot that hold the WAL. You can run below query to confirm and drop this slot should fix the checkpoint failure issue.

> Ref: Troubleshoot Replication Lag



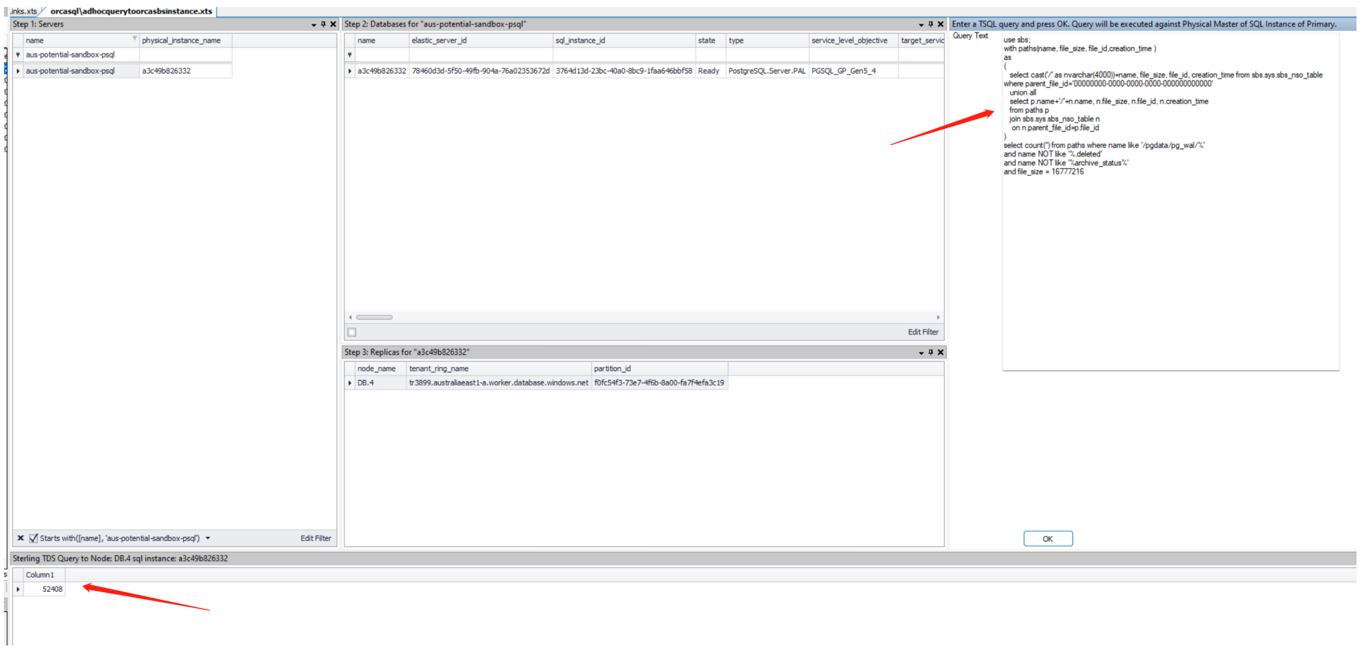
slot_name	active	wal_sender_state	application_name	slot_type	count_	min_PreciseTimeStamp	max_PreciseTimeStamp
airbyte_slot_uat	False			logical	7557	2022-04-06 23:01:04.6786639	2022-05-04 20:14:19.9713121

```
cluster('sqlazureau2.kusto.windows.net').database('sqlazure1').
MonDmPgSqlReplicationStatsPrimary
| where LogicalServerName == '{servername}'
| order by PreciseTimeStamp desc
| summarize count(), min(PreciseTimeStamp), max(PreciseTimeStamp) by slot_name, active, wal_sender_state,
| order by max_PreciseTimeStamp desc
```

Depends on WAL and .snap files, it would still need some time to consume and perform a successful checkpoint. Please refer to below to see how to check

- How to check number of WAL files: In XTS -> ***adhocquerytoorcasbsinstance.xts***, run below

```
use sbs;
with paths(name, file_size, file_id,creation_time )
as
(
select cast('/' as nvarchar(4000))+name, file_size, file_id, creation_time from sbs.sys.sbs_nso_tabl
union all
select p.name+'/'+n.name, n.file_size, n.file_id, n.creation_time
from paths p
join sbs.sys.sbs_nso_table n
    on n.parent_file_id=p.file_id
)
select count(*) from paths where name like '/pgdata/pg_wal/%'
and name NOT like '%.deleted'
and name NOT like '%archive_status%'
and file_size = 16777216
```



.inks.xts / orcasql\adhocquerytoorcasbsinstance.xts

Step 1: Servers

name	physical_instance_name
aus-potential-sandbox-psql	
aus-potential-sandbox-psql	a3c49b826332

Starts with([name], 'aus-potential-sandbox-psql')

Step 2: Databases for "aus-potential-sandbox-psql"

name	elastic_server_id	sql_instance_id	state	type	service_level_objective	target_servic
a3c49b826332	78460d3d-5f50-49fb-904a-76a02353672d	3764d13d-23bc-40a0-8bc9-1faa646bbf58	Ready	PostgreSQL.Server.PAL	PGSQL_GP_Gen5_4	

Step 3: Replicas for "a3c49b826332"

node_name	tenant_ring_name	partition_id
DB.4	tr3899.australiaeast1-a.worker.database.windows.net	f0fc54f3-73e7-4f6b-8a00-fa7f4efa3c19

Enter a TSQL query and press OK. Query will be executed against Physical Master of SQL Instance of Primary.

Query Text

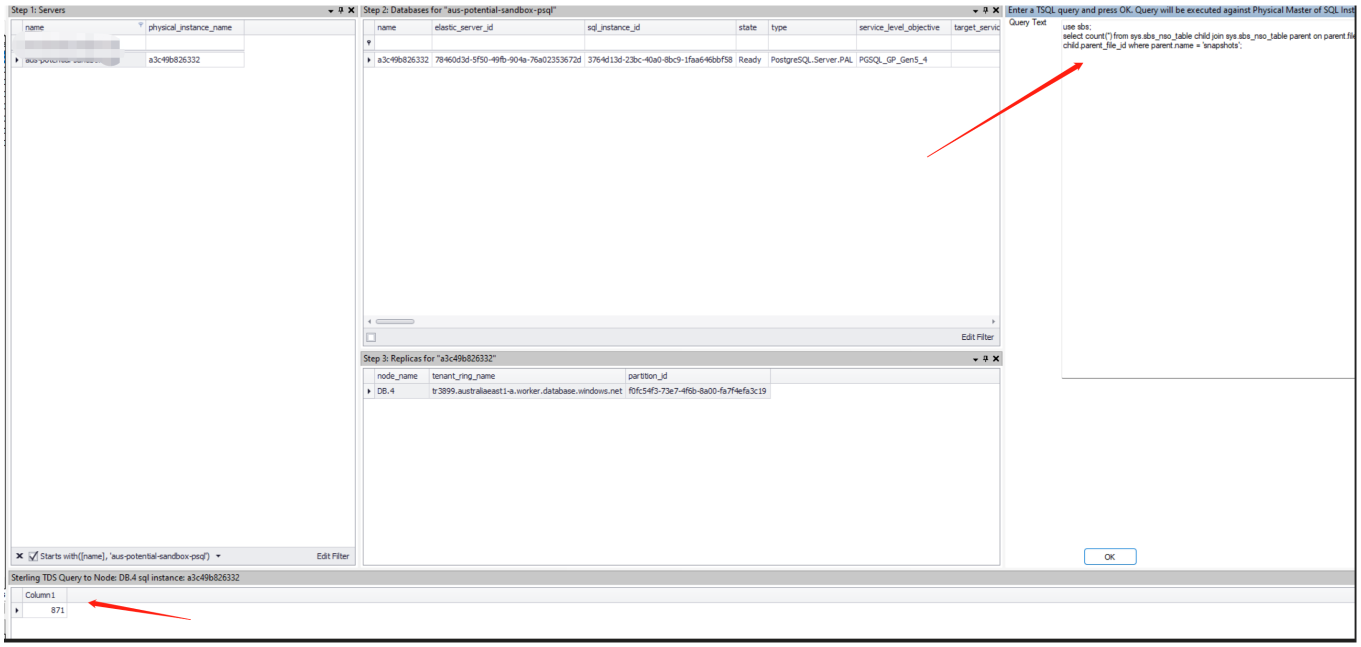
```
use sbs;
with paths(name, file_size, file_id,creation_time )
as
(
   select cast('/' as nvarchar(4000))+name, file_size, file_id, creation_time from sbs.sys.sbs_nso_table
where parent_file_id='00000000-0000-0000-0000-000000000000'
   union all
   select p.name+'/'+n.name, n.file_size, n.file_id, n.creation_time
   from paths p
   join sbs.sys.sbs_nso_table n
    on n.parent_file_id=p.file_id
)
select count(*) from paths where name like '/pgdata/pg_wal/%'
and name NOT like '%.deleted'
and name NOT like '%archive_status%'
and file_size = 16777216
```

Sterling TDS Query to Node: DB.4 sql instance: a3c49b826332

```
Column1
  52408
```

- How to check number of Snap files: In XTS -> ***adhocquerytoorcasbsinstance.xts***, run below

```
use sbs;
select count(*) from sys.sbs_nso_table child join sys.sbs_nso_table parent on parent.file_id = child
```


Step 1: Servers

	name	physical_instance_name
	[redacted]	
▸	[redacted]	a3c49b826332

✕ ✓ Starts with([name], 'aus-potential-sandbox-psql') Edit Filter

Step 2: Databases for "aus-potential-sandbox-psql"

	name	elastic_server_id	sql_instance_id	state	type	service_level_objective	target_servic
▸	a3c49b826332	78460d3d-5f50-49fb-904a-76a02353672d	3764d13d-23bc-40a0-8bc9-1faa646bbf58	Ready	PostgreSQL.Server.PAL	PGSQL_GP_Gen5_4	

Edit Filter

Step 3: Replicas for "a3c49b826332"

	node_name	tenant_ring_name	partition_id
▸	DB.4	tr3899.australiaeast1-a.worker.database.windows.net	f0fc54f3-73e7-4f6b-8a00-fa7f4efa3c19

Enter a TSQL query and press OK. Query will be executed against Physical Master of SQL Inst

Query Text

```
use sbs;
select count(*) from sys.sbs_nso_table child join sys.sbs_nso_table parent on parent.file
child.parent_file_id where parent.name = 'snapshots';
```

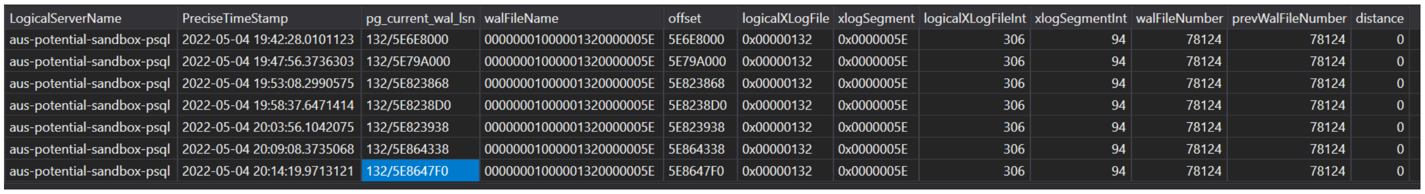
Sterling TDS Query to Node: DB.4 sql instance: a3c49b826332

	Column1
▸	871

How to estimate the time to complete the recovery

1. From the information above(How to confirm if sever is in a recovery mode), you should be able to get the LSN the most recent checkpoint performed

2. Run below query to get the most recent LSN being generated before restart

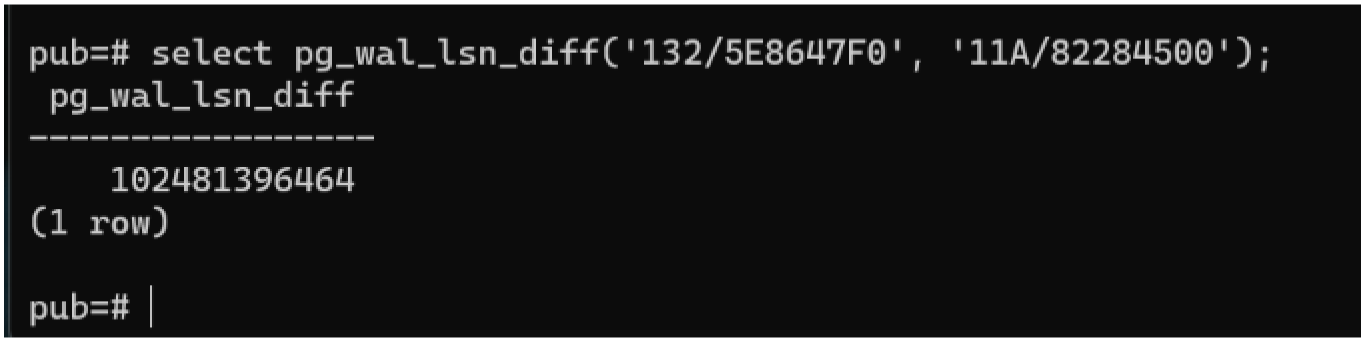


LogicalServerName	PreciseTimeStamp	pg_current_wal_lsn	walFileName	offset	logicalXLogFile	xlogSegment	logicalXLogFileInt	xlogSegmentInt	walFileNumber	prevWalFileNumber	distance
aus-potential-sandbox-psql	2022-05-04 19:42:28.0101123	132/5E6E8000	00000001000001320000005E	5E6E8000	0x00000132	0x0000005E	306	94	78124	78124	0
aus-potential-sandbox-psql	2022-05-04 19:47:56.3736303	132/5E79A000	00000001000001320000005E	5E79A000	0x00000132	0x0000005E	306	94	78124	78124	0
aus-potential-sandbox-psql	2022-05-04 19:53:08.2990575	132/5E823868	00000001000001320000005E	5E823868	0x00000132	0x0000005E	306	94	78124	78124	0
aus-potential-sandbox-psql	2022-05-04 19:58:37.6471414	132/5E8238D0	00000001000001320000005E	5E8238D0	0x00000132	0x0000005E	306	94	78124	78124	0
aus-potential-sandbox-psql	2022-05-04 20:03:56.1042075	132/5E823938	00000001000001320000005E	5E823938	0x00000132	0x0000005E	306	94	78124	78124	0
aus-potential-sandbox-psql	2022-05-04 20:09:08.3735068	132/5E864338	00000001000001320000005E	5E864338	0x00000132	0x0000005E	306	94	78124	78124	0
aus-potential-sandbox-psql	2022-05-04 20:14:19.9713121	132/5E8647F0	00000001000001320000005E	5E8647F0	0x00000132	0x0000005E	306	94	78124	78124	0

```
cluster('sqlazureau2.kusto.windows.net').database('sqlazure1').
MonDmPgSqlXlogFileCount
| where LogicalServerName == "{servername}"
| extend index = indexof(pg_current_wal_lsn, '/', 0, -1, 1)
| extend offset = substring(pg_current_wal_lsn, index + 1)
| extend len = 8 - strlen(offset)
| extend walFileName = strcat('00000001', strrep('0', 8-index), substring(pg_current_wal_lsn, 0,index), '
| extend logicalXLogFile = strcat('0x',substring(walFileName, 8, 8))
| extend logicalXLogFileInt = toint(logicalXLogFile)
| extend xlogSegment = strcat('0x',substring(walFileName, 16, 8))
| extend xlogSegmentInt = toint(xlogSegment)
| extend walFileNumber = logicalXLogFileInt * 255 + xlogSegmentInt
| order by PreciseTimeStamp asc
| project  LogicalServerName, PreciseTimeStamp, pg_current_wal_lsn, walFileName, offset, logicalXLogFile,
```

3. Run below SQL function in a PostgreSQL interface to get the differences in WAL size(in bytes). Normally, 60GB would take 5 hours.

```
select pg_wal_lsn_diff('11A/82284500', '129/92D05F78');
```



```
pub=# select pg_wal_lsn_diff('132/5E8647F0', '11A/82284500');
 pg_wal_lsn_diff
-----------------
    102481396464
(1 row)

pub=#
```

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