

Migrate to Azure Database for PostgreSQL Flexible Server Using Warp Migration Tool

Warp Migration Tool can be used to migrate a source PostgreSQL database (cloud, on-premises) to PostgreSQL PaaS offering on Azure - including Flexible Server, Single Server and Hyperscale (Citius). The tool is a large-scale command-line data migration capability which allows for effective and fast big data migrations. This tool takes care of synchronizing pre-existing changes in the source and destination data and tracks any updates, it can do this by taking advantage of the use of WAL (Write-ahead logging) to achieve reliable and stateless replication of data.

Table of Contents

Table of Contents

<i>Migrate to Azure Database for PostgreSQL Flexible Server Using Warp Migration Tool</i>	<i>1</i>
Why Migrate?	1
Setting Up Prerequisites	2
Example flow of the migration for a sample Database from Single server to Flexible server	2
Actions To Take During Cutover	10
Warp Features	10
Top Switches for Warp Migration Tool	12
Common Issues and Error Messages	13
Current Limitations	15
Multi-Processing Pre-Release.....	15
FAQ	15
Questions	17

Why Migrate?

Azures recently launched [Flexible Server](#) is designed to be a lot more customizable in terms of optimizations and features.

- [Migrate on premises to the cloud on Azure PostgreSQL Flexible Server.](#)
- Migrate from PostgreSQL Single Server to PostgreSQL Flexible Server.
- Migrate from PostgreSQL Single Server to Hyperscale (Citius).

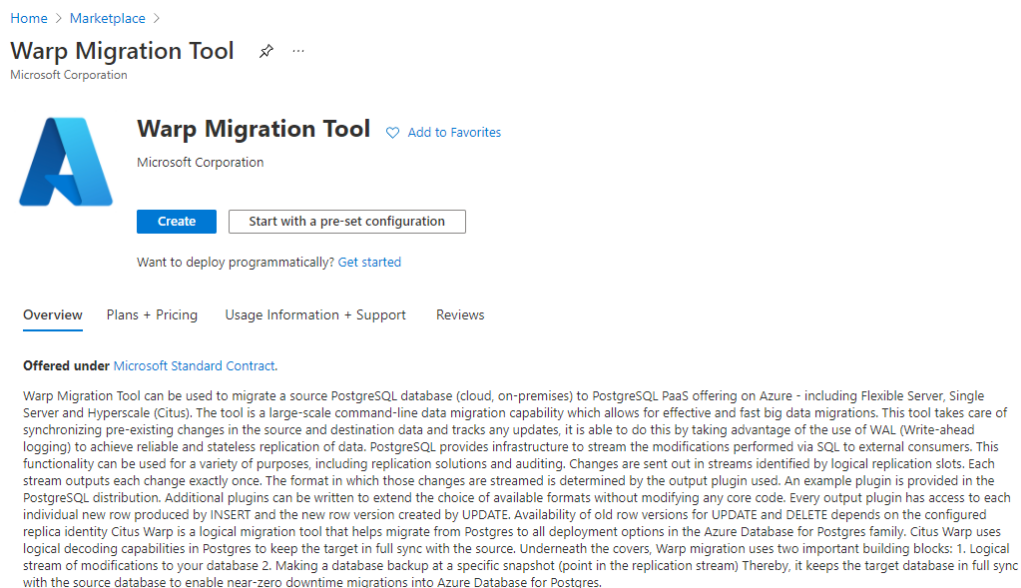
Setting Up Prerequisites

- Setup the Warp Migration Tool – This is covered in the below section.
- Verify that the source database version is 9.4 or higher. This is necessary because warp uses a logical decoding feature in Postgres and is supported only since the 9.4 version.
- Migrate schema from the source to the target – This is covered in the below section.
- Ensure that there are PRIMARY KEY constraints on all the tables that are to be migrated using warp. Any table without PRIMARY KEY constraint can be excluded from the migration using -T flag (reference the top switches section).
- Enable logical replication on the source database. If the source is single server [use this document](#). If you are migrating from on-premises or other cloud, ensure that Postgres setting wal_level is set to logical. Note that changing this setting requires a restart, so you need to plan maintenance time on the source/production database.
- Create Destination Server – Azure Database for PostgreSQL Server
 - Refer to [Quickstart: Create server - Azure portal - Azure Database for PostgreSQL - Flexible Server | Microsoft Docs](#)
- Update destination database server parameter **session_replication_role** to REPLICA in Azure portal, to disable foreign keys and triggers.

Example flow of the migration for a sample Database from Single server to Flexible server

The first step is to setup warp (Create a VM) that will be hosting the migration. Below are the step-by-step instructions on how to setup warp to have a smooth and successful data migration.

- Go to [Warp Migration Tool](#) in Azure Marketplace and Select **Create**.



The screenshot shows the Azure Marketplace page for the Warp Migration Tool by Microsoft Corporation. The page includes a navigation bar with 'Home > Marketplace >', the tool's name 'Warp Migration Tool' with a star icon, and the publisher 'Microsoft Corporation'. Below this is a large blue 'A' logo and a 'Create' button. A link 'Start with a pre-set configuration' is also visible. A section titled 'Offered under Microsoft Standard Contract.' contains a detailed description of the tool's capabilities, including its use for migrating PostgreSQL databases to Azure PaaS offerings like Flexible Server, Single Server, and Hyperscale (Citius). The description highlights features like logical replication, WAL (Write-ahead logging), and the ability to stream modifications to external consumers.

- In the **Basics** tab, under **Project details**, make sure the correct subscription is selected and then choose to **Create new or Use an existing** resource group.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Pay-As-You-Go
Resource group *	(New) myResourceGroup

[Create new](#)

- Under **Instance details**, type an appropriate name for the **Virtual machine name** and choose your **Region**. Choose appropriate **Size** based on the number of threads for parallel dump/restore will be used (Our recommendation is to choose a VM which has at least 4 Vcores). For the **Image** leave the defaults. *Note: Choose a region which is closest to either the source or target database. For example, if you are migrating to a Flexible Server in East US, create the marketplace VM in East US.*

Instance details

Virtual machine name *	myVM
Region *	(US) East US
Availability options	Availability zone
Availability zone *	1
Image *	caroline_warp_dev/caroline_warp_release_test_2/latest - Gen2
	See all images
Azure Spot instance	<input type="checkbox"/>
Size *	Standard_B4ms - 4 vcpus, 16 GiB memory (\$121.18/month)
	See all sizes

- Under **Administrator account**, Select either SSH public key or Password. **Must provide username azureuser**. If password option is selected the password must be at least 12 characters long and meet the [defined complexity requirements](#).

Administrator account

Authentication type	<input type="radio"/> SSH public key <input checked="" type="radio"/> Password
Username *	azureuser
Password *
Confirm password *	

- Under **Inbound port rules**, choose **None**. *Note: Must authorize port 22 from specific IP addresses for user to SSH into the instance. This will be covered in a later section.*

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ⓘ

- ☒ None
☐ Allow selected ports

Select inbound ports

Select one or more ports

i All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

- Leave the remaining defaults and then select the **Next: Disks >** button at the bottom of the page.

Licensing

License type *

If you are using a RedHat or SLES image, you may be eligible for the Azure Hybrid Benefit and can save money on the license costs. [Learn more](#) about this benefit and how to enable it using Azure CLI for custom images from snapshots and Shared Image Gallery.

Review + create

< Previous

Next : Disks >

- Depending on the size of the database, a bigger disk will need to be attached to be used for dump/restore to avoid running out of disk space. On the **Disks** tab select either “create and attach a new disk” or “Attach an existing disk”. For further instruction refer to [Attach a data disk to a Linux VM - Azure Virtual Machines | Microsoft Docs](#).

Basics Disks Networking Management Advanced Tags Review + create

Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional data disks. The size of the VM determines the type of storage you can use and the number of data disks allowed. [Learn more](#)

Disk options

OS disk type * ⓘ

Premium SSD (locally-redundant storage)

Encryption type *

(Default) Encryption at-rest with a platform-managed key

Enable Ultra Disk compatibility ⓘ

☐

Data disks

You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.

LUN	Name	Size (GiB)	Disk type	Host caching
Create and attach a new disk		Attach an existing disk		

- Select the **Next: Networking >** button at the bottom of the page.

Review + create

< Previous

Next : Networking >

- In the **Networking** tab, under **Network Interface**, select a virtual network that has access to the source and target database servers or create new. Leave all other defaults.

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network * ⓘ

sai-warp-vnet

Create new

Subnet * ⓘ

default (10.0.0.0/24)

Manage subnet configuration

Public IP ⓘ

(new) myVM-ip

Create new

NIC network security group ⓘ

☐ None
 ☒ Basic
 ☐ Advanced

Public inbound ports * ⓘ

☒ None
 ☐ Allow selected ports

Select inbound ports

Select one or more ports

ⓘ All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

Accelerated networking ⓘ

☐

The selected image does not support accelerated networking.

- Leave the remaining defaults and then select the **Review + create** button at the bottom of the page.

Review + create

< Previous

Next : Management >

- After validation runs, select the **Create** button at the bottom of the page.

Create

< Previous

Next >

Download a template for automation

- After deployment is complete, select **Go to resource**.

✓ Your deployment is complete



Deployment name: CreateVm-caroline_warp_release_test_2-202107...
Subscription: [Orcas PM team](#)
Resource group: [karla-rg](#)

Start time: 7/30/2021, 4:50:34 PM
Correlation ID: fb923196-6eef-441d-b12f-23223bde5ac0

✓ Deployment details [\(Download\)](#)

^ Next steps

[Go to resource](#)

- Select the **Networking** tab, then select **Add inbound port rule** to add the correct IP address to access the Virtual Machine.



Karlatestwarp716 | Networking ...

Virtual machine

Search (Ctrl+ /)

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Networking

Connect

Disks

Size

Security

Advisor recommendations

Attach network interface Detach network interface

Virtual network/subnet: [sai-warp-vnet/default](#) NIC Public IP: [REDACTED]
NIC Private IP: [REDACTED] Accelerated networking: **Disabled**

Inbound port rules

Outbound port rules


Application security groups

Network security group [Karlatestwarp716-nsg](#) (attached to network interface: [karlatestwarp716457](#))
Impacts 0 subnets, 1 network interfaces

[Add inbound port rule](#)

Priority	Name	Port	Protocol
100	karlalaptop	Any	Any
65000	AllowVnetInBound	Any	Any
65001	AllowAzureLoadBala...	Any	Any
65500	DenyAllInBound	Any	Any

- Fill in appropriate fields, be sure to **authorize port 22** here as pointed out in previous instruction and select **Add** when finished. *Note: to see your IP address, run command **curl ifconfig.me** in your local machine.*



Add inbound security rule

KarlatestWarp730-nsg
×

Source ⓘ

IP Addresses

Source IP addresses/CIDR ranges * ⓘ

10.0.0.0/24 or 2001:1234::/64

Source port ranges * ⓘ

*

Destination ⓘ

Any

Service ⓘ

Custom

Destination port ranges * ⓘ

22

Protocol

☒ Any

☐ TCP

☐ UDP

☐ ICMP

Add Cancel

Now that warp is setup (VM was created), we SSH into machine to run warp and begin data migration.

- Once inbound port rule is added select the **Connect** tab and select **SSH**. Here you will need to copy command `SSH azureuser@PublicIPAddress` on local machine.

4. Run the example command below to connect to your VM.

```
ssh -i <private key path> karla@
```

- Connect via SSH with client on local machine. If VM was created with password (like example below) enter password to access the VM.

```
kesco1@LAPTOP-0PM4COPH:~$ ssh karla@
karla@: 's password:
```

- If a new disk was created and attached during initial setup, you will need to partition and mount the disk first before it can be used. Instructions to mount the disk are given [here](#). Ensure this mounted disk is used as the destination for the database dump when you run Warp.
- Run command `ls` to view that warp files are in place.

```
azureuser@Karlatest723:~$ ls
warp
```

- Run command `cd warp/` to access directory.

```
azureuser@Karlatest723:~$ cd warp/
azureuser@Karlatest723:~/warp$
```

- Run command `./citus_warp --setup` to begin entering source and destination information. *Note: For full list of top switches refer to Top Switches for Warp Migration Tool Section.*

```
azureuser@Karlatest723:~/warp$ ./citus_warp --setup
Custom input file exists, are you sure you want to overwrite it? [y/n/v]: y
Source Host: 
Leave empty to keep port "5432" as your source port.
Source Port: 
Source Username: 
Source Password: 
Source Database: 
Leave empty to keep SSL Mode as "required".
Source SSL Mode [<empty>/False]: 
Destination Host: 
Leave empty to keep port "5432" as your destination port.
Destination Port: 
Destination Username: 
Destination Password: 
Destination Database: 
Leave empty to keep SSL Mode as "required".
Destination SSL Mode [<empty>/False]: 
azureuser@Karlatest723:~/warp$
```

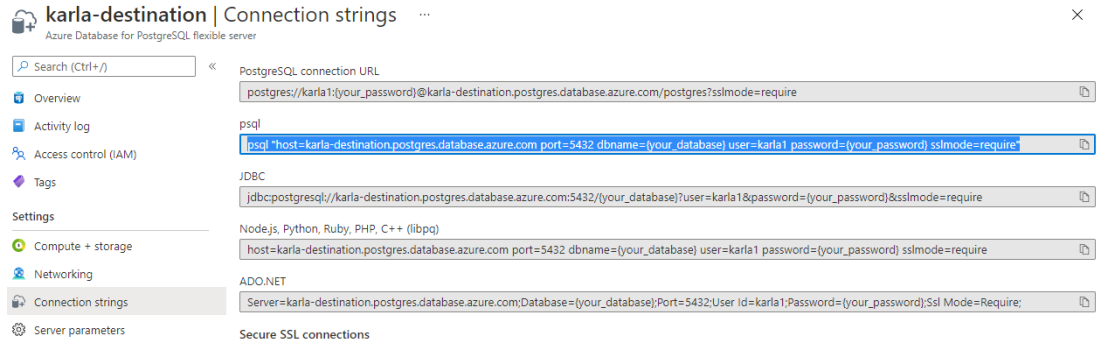
- Run command `curl ifconfig.me` and add the Warp IP address to both source and destination servers through the Azure Portal.
- Run command `pg_dump -h [Source host name] --username=[admin username] -d [Database name] --schema-only > schema.sql`

```
azureuser@Karlatest723:~/warp$ pg_dump -h [Source host name] --username=[admin username] -d sportsdb_sample --schema-only > schema.sql
Password:
```

- D `DB_name.sql` to view entire database schema.

```
carolineliu@karlawarpcarolinetest716:~/citus_warp_dir$ ls
README_TEST.md citus_warp inputs.txt inputs_no_overwrite.txt inputs_overwrite.txt schema.sql test_suite.sh
carolineliu@karlawarpcarolinetest716:~/citus_warp_dir$ cat schema.sql
```

- In the Azure Portal under the destination server select the **connection strings** tab and copy the psql command (highlighted below).



- Run command with appropriate inputs and create target database. This creates an empty target database.

```
azureuser@Karlatest723:~/warp$ psql "host=karla-destination.postgres.database.azure.com port=5432 dbname=
postgres user=karla1 sslmode=require"
Password for user karla1:
psql (13.3 (Ubuntu 13.3-1.pgdg20.04+1), server 12.7)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=> create database sportsdb_sample;
```

- Run `\q` command to quit, then run command `ls` to ensure schema is copied.

```
postgres=> create database sportsdb_sample1;
CREATE DATABASE
postgres=> \q
azureuser@Karlatest723:~/warp$ ls
citus_warp  schema.sql
```

- Run command `psql "host=[Enter server name] port=0000 dbname=[your_database] user=[Enter username] password=[your_password] sslmode=require" < schema.sql` to import schema to the target.

```
azureuser@Karlatest723:~/warp$ psql "host=karla-destination.postgres.database.azure.com port=5432 dbname=
sportsdb_sample1 user=karla1 sslmode=require" < schema.sql
Password for user karla1:
```

- Run command `./citus_warp` to run warp (To see all available options reference top switches section for Warp) *Note: This option allows for only a single large table, for initial load warp runs `pg_dump` and `pg_restore` simultaneously and pipes `pg_dumps` output to `pg_restore`. As `pg_dump/pg_restore` are single threaded at a single table level, piping dump/restore is the most optimal.*

```

azureuser@karlatest723:~/warp$ ./citus_warp
.default_custom_config.yaml config file not found, the following exist:
1) .karla.yaml

To select a config file, type the file number or filename; to exit, type "n": 1
Config file found: .karla.yaml
Starting Warp...
2021-07-23 21:33:06.146 UTC: Creating replication slot on source...
Run TRUNCATE on destination: postgres://karla1:Justin%21191198@karla-destination.postgres.database.azure.com:5432/sports
db_sample?sslmode=require
Please type yes or no and then press enter:
yes
2021-07-23 21:33:10.249 UTC: Truncating destination tables...
2021-07-23 21:33:11.477 UTC: Running initial backup...
2021-07-23 21:33:21.898 UTC: Creating replication origin on destination...
2021-07-23 21:33:21.987 UTC: Started replication
2021-07-23 21:33:27.100 UTC: Stats: 0.0 TX/s, 0.0 inserts/s, 0.0 updates/s, 0.0 deletes/s, replication: 0 MB lag | LSN:
0/0 | 0.0 messages/s
2021-07-23 21:33:32.210 UTC: Stats: 0.0 TX/s, 0.0 inserts/s, 0.0 updates/s, 0.0 deletes/s, replication: 0 MB lag | LSN:
0/0 | 0.0 messages/s
2021-07-23 21:33:37.304 UTC: Stats: 0.0 TX/s, 0.0 inserts/s, 0.0 updates/s, 0.0 deletes/s, replication: 0 MB lag | LSN:
0/0 | 0.0 messages/s

```

- Another option to run Warp If you have multiple large/medium sized tables, you can use the parallel dump/restore option that warp provides. Run command `./citus_warp -parallel-dump=n -parallel-restore=m -tmp-dir=<dump_folder>` Command parallelizes the dump and restore using n and m threads respectively. This helps better utilize resources (network bandwidth/compute) and improve performance of initial load.
- What to Expect?
 - Warp performs initial load using pg_dump/pg_restore. This can run for hours based on the size of the database.
 - Once the initial load completes, warp gets into replication state, where it replays all the ongoing changes until both source and target are caught up.

Actions To Take During Cutover

- Make sure Warp replication lag is 0.
- Enter maintenance mode on the application.
- Ensure that all background/maintenance processes are stopped.
- Wait for existing sessions to expire or terminate them directly.
- Ensure that Warp is not seeing any activity.
- Restart Warp with batch size 1 and ensure replication lag is still 0 - stop warp command in replication phase and add these flags `--batch-statements=1 --batch-transactions=1`
- Perform a test update on source database and verify that change is reflected in destination
- Change the passwords to the old database to break new connections
- Stop Warp
- Fix sequences
- Revert destination database server parameter ***session_replication_role*** value to ORIGIN
- Point your application at Destination database
- Perform comprehensive application testing
- Let the users back in! See [here for sequence fixes](#).

Warp Features

- **Custom configuration setup**

To perform migrations from a source to a destination server on a subset of tables in a database,

Warp relies on a connection string to each. However, both the source and destination servers take in multiple similar arguments and when performing multiple or simultaneous migrations, it can get complicated. To streamline the process, Warp now supports configuration files.

This can be performed with the `--setup` flag; it prompts for the source and destination server parameters, respectively, storing them in the specified configuration file. By default, it is stored in `.default_custom_config.yaml`. However, you may specify other filenames with the `-config` flag.

If you find that you've made a mistake in setting up a configuration file, you may fix it in two ways. By running `-setup` again with the specified filename, you may rewrite the file (you'll be prompted with yes [y], no [n], or v [visualize]). You can also go in to manually edit the file with vim filename; vim has been preconfigured for ease of usage.

For any future Warp operations requiring these parameters, pass in the configuration file with `-config`. If a configuration file is not specified, Warp will take the default filename; if it hasn't been set up, you'll be prompted to set one up.

To bypass configuration files altogether, a source and destination connection URL will have to be provided respectively, with `-source (-s)` or `-destination (-d)`.

- **Multithreading for one-server-to-one-destination dump/restore**

Migrations typically involve large quantities of data; dumps and restores often take a long time in those cases, as Warp must cycle through each entry in a table in a database. To ameliorate this, Warp supports multi-threaded parallel dumps and restores within a database (for table support, see below).

Parallel dumps and restores can be achieved with the `--parallel-dump=n` and `--parallel-restore=m` flags where $n = m$ is not necessarily true. n refers to the number of threads optimal for the source server and m is that of the destination server. Often, $n < m$, particularly if the source is a single server and the destination is a flexible server (or anything else) as the infrastructure in the destination servers can generally support more parallelism than the source.

<p>Best Practice: An ideal number of dump threads would be based on the CPU utilization and available CPU on the source, you can choose the number of dump threads in a way that it doesn't affect the source database; the goal is for the dump to avoid affecting production load on the database. For the destination, because there should be little to no production, a good number of restore threads is just the number of virtual cores, but no more than 12.</p>
--

For parallel dump and restores, a temporary directory to store dump data must be created; this helps log the state of compressed data across the multiple threads. It is recommended that you make a new temporary directory for the data dump which can be passed in through `-tmp-dir` using either the full or relative file path.

- **Persistence across server crashes**

With such larger production data transfers, crashes are almost guaranteed to happen. There are three main segments across which a crash may occur: the initial dump phase, restore phase, and replication phase.

Starting from the back, a crash in the replication phase is the simplest to handle. To continue, Warp just needs to be rerun for that instance. The expected behavior that this stage is that replication will continue from where it last left off. This is the case for both single and multi-threaded instances of Warp.

In the instance that Warp crashes during the dump phase, for single-threaded instances of Warp, the replication slot must first be cleaned with `--clean --slot=slotname` (default being `citus_warp`) before restarting Warp. For parallel instances of dump, the slot must be cleaned first; subsequently, the dump directory must be removed with `rm -rf <tmp_dir>/dump` before restarting Warp.

In the instance that Warp fails during the restore phase, for restoration in single-threaded instances, the slot must be cleaned and Warp restored, as in dump failures. In multi-threaded instances, we notice a few things. We know that the dump phase has completed, which can be verified in the `<tmp_dir>/dump` folder with compressed data dump files; we don't want to go through the intensive dump process again. Instead, we want to focus on restoring, then syncing the databases. To do this, we first run Warp with `--restore-only` first which does not automatically remove the dump directory; thus, we must manually remove it. Then, we run Warp with `--skip-initial-sync` to continue Warp regularly.

- **Batch transactions/statements**

For better optimizations and efficiency, Warp also supports batch statements and transactions! By default, there is only ever one statement or transaction that is batched together; however, it can easily support up to a few thousand for each. They can respectively be set with `--batch-statements` and `---batch-transactions` respectively. Batching helps avoid too many network round trips to the target thereby improving replication throughput.

<p>Best Practice: For workloads with high throughput transaction rates (ex: 1000s of TPS), we have seen <code>---batch-statements=2000</code> and <code>--batch-transactions=500</code> to give maximum performance. Once in replication phase, you can investigate <code>messages/s</code> metric to see what throughput you are getting and tune these flags accordingly.</p>
--

Top Switches for Warp Migration Tool

--setup

Sets up configuration files (should be first command run). To set up only the default, run as default; to set up your own configuration file.

-T, --exclude-table value

Do NOT dump, restore, and replicate the named table(s).

-t, --table value

dump, restore and replicate the named table(s) only

--sync-schema

Include schema in the initial backup process (this will remove any existing objects on the target).

--clean-full

Performs a full clean of warp; removes configuration settings, files, and directories; cleans replication slot on source and replication origin on destination. Should be used when complete with a set of warp runs for source/destination pairing.

--clean

Cleans replication slot on source, and replication origin on destination (run this after you don't need replication anymore).

--v

Gives detailed outputs of each command.

--help

For a full list of top switches in Warp Migration Tool.

Common Issues and Error Messages

2021-07-30 13:41:52.097 UTC: Truncating destination tables...

2021-07-30 13:42:33.838 UTC: Running initial backup dump (8 parallel jobs)...

2021-07-30 15:18:19.384 UTC: Running initial backup restore (8 parallel jobs)...

2021-07-30 20:51:01.033 UTC: Creating replication origin on destination...

write tcp 172.18.0.4:33618->20.102.24.125:5432: write: connection timed out

Above shows that backup and restore are completed and during replication phase there was a connection timeout. You can continue from replication phase by following below steps:

- Connect to the source database via psql and confirm that the replication slot exists. This can be done via running the below command
 - `SELECT * from pg_replication_slots;`
- Connect to the target database and create the lsn table that would help keep state (last committed source lsn) during replication. This is necessary for crash recovery purposes during replication crashes/failures.
 - `CREATE TABLE IF NOT EXISTS lsn(lsn text);`
- Restart warp by adding the flag `--skip-initial-sync`. Keep other flags as before.
 - `./citux_warp --skip-initial-sync <other_flags>`

Sometimes lag doesn't reduce. This could be because you are using large values for batch-statements and batch-transactions. In such a scenario:

- Stop warp (cntrl+c)
- Start warp with batch-statements and batch-transactions set to 1. Keep other flags as before.
 - `./citus_warp --batch-statements=1 --batch-transactions=1`
- This typically happens when lag is less (<10-15GB) and is close to catchup. Hence this step of changing batch setting to 1 is a part of the cutover checklist section.

2021-01-22 07:07:16.942 UTC: Truncating destination tables...

2021-01-22 07:07:36.767 UTC: Running initial backup dump (10 parallel jobs)...

2021-01-26 06:55:00.989 UTC: Running initial backup restore (16 parallel jobs)...

pg_restore: error: error returned by PQputCopyData: SSL SYSCALL error: EOF detected

pg_restore: error: a worker process died unexpectedly

ERROR: Initial backup failed: Error restoring data: exit status 1

This means that warp had issues during restore process. The good news is that you wouldn't need to perform dump again. However, you need to trigger warp for restore and replication separately using the below steps.

- Restart warp with --restore-only flag. Keep other flags as before.
 - `./citus_warp --restore-only`
- After restore is done, connect to the target database, and create the lsn table that would help keep state (last committed source lsn) during replication. This is necessary for crash recovery purposes during replication crashes/failures.
 - `CREATE TABLE IF NOT EXISTS lsn(lsn text);`
- Remove --restore-only flag and restart warp by adding the flag --skip-initial-sync. Keep other flags as before.
 - `./citus_warp --skip-initial-sync <other_flags>`
- Once both databases are caught up and cutover is completed, make sure to remove the dump (in the tmp-dir). If restore fails, warp doesn't remove the dump directory.

pg_dump: error: a worker process died unexpectedly

ERROR: Initial backup failed: Error dumping data: exit status 1

- This means that the dump process failed. Unfortunately, warp must be restarted from scratch.
- First clean all the previous artifacts that warp generated. Keep other flags as before.
 - `./citus_warp --clean <other_flags>`
- If you have provided the path to a directory where you want to store the dump (using --tmp-dir flag), go to that directory, and remove the dump folder.
- Restart warp with the same flags as before

Current Limitations

- **Multithreading across a singular table**

Although multithreading dumps and restores across multiple tables is supported with Warp, currently dumps, and restores across a table is still coarse and is restricted to one thread per table. This is in development, and you can expect future versions of Warp to include better support for this feature.

- **Including/excluding tables**

Within a database, Warp can perform migrations for most tables and their respective features. However, one major limitation is caused by the exclusion of certain tables from the dump and restore process. If an excluded table is referenced as a foreign key in an included table, it can cause potential migration issues and must be handled separately.

- **Database/parameter naming conventions**

This is less of a limitation of Warp and more so of PostgreSQL, upon which rely. Postgres does not support encodings of characters outside of the standard ASCII encodings; this means any extra UTF-8 encodings and further are not supported, as well as extended ASCII encodings. If your server, database, username, password, or host name includes any of these characters, they need to be changed before a connection can be established.

Multi-Processing Pre-Release

- If your migration requires multiple instances of Warp to be run, you may find the beta release of Warp useful as it supports multiprocessing within Warp as well as a logging system. Running Warp with multiple configuration files and replication slots allows for multiple instances of Warp to be run at once, and for their individual states to be logged into `.<config_filename>.log` files. To enable this, simply run Warp with `--config=filename1,filename2,...,filename` and `--slot=slot1,slot2,...,slotn` with the filenames and slot names passed in as a 1-1 relationship.

You may view the log files in whichever methods you'd like, but two recommended methods are through vim, and subsequently the edit command to see updates or to use `cat filename`, to display it in stdout (terminal output).

FAQ

- **How do I see all my options when running Warp?**

`./citus_warp --help` gives all possible flags and their respective behaviors.

- **Why does Warp help keep showing up every time I try to run a command?**

If you run Warp with incorrect or unexpected arguments into a flag, it will display what `--help` shows. Double check the format of the flags and the expected arguments, if any.

- **I'm sure my configuration file is correct, why is Warp not working?**

There can be multiple reasons for this. One common reason is that your machine is unable to connect to your servers. To test this, use the `--testing` flag to verify connection stability and mobility.

Additionally, make sure you're using the correct configuration file. To generate a list, run Warp without any configuration fields and follow the prompt to confirm.

If it's still not working, verify your credentials are correct by opening the respective configuration file and manually check the spelling of the fields. No manual encoding is required at this stage.

If it is continuing not to work, contact Warp support at Microsoft.

- **Why does Warp keep telling me to “clean” my replication slot and what does that mean?**

Replication slots are created when a connection can be established across the machine across both the source and destination servers. Once this is established, a replication slot is created on the destination server and a replication origin is created on the source server. These two points act as communication foci for when replication is under way. However, Warp needs these names to be unique across processes and instances, so if we never get to the replication phase, the two points are unused. When we restart Warp, it's looking for a unique slot name, so if we don't clean the slot, Warp won't recognize the process as unique and start properly.

- **How can I view my previous Warp outputs?**

TODO When using the beta Warp executable, log files can be viewed for each configuration file. For the standard Warp executable, logs can (will be) enabled through the --log flag. By default, logs will be stored in .<config_filename>.log files; if a separate logfile is passed in through -log, logs for that Warp instances will be stored there instead.

- **How can I speed up the dump process? It's taking a long time.**

If your database has as large number of tables and the server can support multi-threading, i.e., there are multiple virtual cores, you can try parallel dumps. However, keep in mind this will not necessarily speed up your dump by n times, with n being the number of parallel dump threads. This can happen if one (or a small number of tables) is/are much larger than most of the tables; because Warp does not support intra-table-level parallelism, the dump will still be rate-limited by your largest tables.

Similarly, if your server does not have enough cores, your dump will be limited to that number of threads. If all cores are being used for the dump, this may also cause latency and support issues on your source server end, so you may not way to allocate all cores to migration purposes. If you see high CPU utilization in source/target databases, you can temporarily scale up the number of virtual cores in the servers during the migration, then scale down after migration is complete.

Another option is to considering scale up storage on source and destination database server to improve IOPs. Make sure to have at least 20% more storage to be cater logical replication slot size.

- **How can I confirm the ongoing replication is correct?**

There are multiple ways of checking ongoing replication; correctness can be difficult to directly validate outside of migration metadata though.

One way of checking to see if replication is occurring is to see if the LSN is changing. To do this, you can either check the logs of the configuration run if logs are enabled (or the terminal standard output if not) to see if when the source server receives updates, that the LSN also changes.

You can also directly access the database and use standard SQL queries to cherry-pick tables' statuses to check, both on the source and destination servers to ensure they're being updated just-in-time correctly.

- **How can I keep Warp running if a process is going to take a significant amount of time?**

We suggest running Warp from inside of a session manager; the recommended one is `tmux` (terminal multiplexer), but `screen` also does the job. Even though Warp is being run from inside a virtual machine, if the connection to the virtual machine is broken, regardless of whether the machine is still up and running or not, any processes from inside the VM will be terminated.

To prevent that, we maintain the necessary processes from within a session. To accomplish this, create a new session, start Warp, and run everything as intended, and when Warp is left to run.

- **What's the process for tearing down Warp?**

If you're done with an instance of Warp, e.g., done a full migration or no longer need the just-in-time replication, you can clean up Warp to various extents.

To clean up just the replication slots, running Warp with the configuration file associated with the slot and `--clean` does the trick. To remove configuration files and the replication slots, as well as Warp setup materials, replacing `--clean` with `--clean-full` does the trick. However, this should be used sparingly as it requires root privileges. Additionally, running Warp with configurations after a full clean will also require initial root privileges.

Questions

- If you have any questions, please email AskAzureDBforPostgreSQL@service.microsoft.com

