Performing a Dump/Restore from one Hyperscale (Citus) cluster to another

Table of Contents

[Create a beefy (ex: 16vCores, 2TB storage) Azure Linux VM 2](#_Toc42680106)

[Install postgres on the VM 2](#_Toc42680107)

[For RHEL/CentOS 2](#_Toc42680108)

[For Ubuntu 2](#_Toc42680109)

[Steps on Source Cluster 2](#_Toc42680110)

[Schema-only Dump 2](#_Toc42680111)

[Separate out primary/unique/foreign keys and indexes 2](#_Toc42680112)

[Get distributed and reference table statements 2](#_Toc42680113)

[Data-only Dump 2](#_Toc42680114)

[Steps on Destination Cluster 3](#_Toc42680115)

[Apply schema 3](#_Toc42680116)

[Distribute and make table referenced 3](#_Toc42680117)

[Data-only restore 3](#_Toc42680118)

[Apply primary/unique/foreign keys and indexes 3](#_Toc42680119)

[Vacuum the database 3](#_Toc42680120)

[Data validation 3](#_Toc42680121)

**Last updated**: 2020.06.10

**Important**: These steps address the public schema only. Tweak the steps to handle other schemas as needed.

# Create a beefy (ex: 16vCores, 2TB storage) Azure Linux VM

Create a beefy (ex: 16vCores, 2TB storage) Azure Linux VM in the same region as the source and destination clusters. If the source and destination are in different regions, create two such VMs, one in source db region and the other in the destination db region.

In the source region, perform “Steps on Source Cluster”, and in the destination region, perform “Steps on Destination Cluster”. Move artifacts (dumps, etc.) that result from “Steps on Source Cluster” to the destination VM via scp. To improve the speed of scp, compress (zip) these artifacts.

# Install postgres on the VM

Use the steps below to install postgres. This is just to get the pg\_dump/pg\_restore utilities. Postgres/Citus will actually be on the PaaS cluster.

## For RHEL/CentOS

curl https://install.citusdata.com/community/rpm.sh | sudo bash

sudo yum install -y citus93\_12

## For Ubuntu

curl https://install.citusdata.com/community/deb.sh | sudo bash

sudo apt-get -y install postgresql-12-citus-9.3

# Steps on Source Cluster

## Schema-only Dump

Run from bash/terminal.

pg\_dump --schema-only --format=plain --no-owner --file=schema.sql --schema=public "host=<host-name> port=5432 dbname=citus user=citus password=<password> sslmode=require"

## Separate out primary/unique/foreign keys and indexes

Open the above schema.sql file and separate all primary/unique/foreign keys and indexes to their own file – name it indexes.sql

This step creates the indexes after data load, which will help greatly improve restore performance.

## Get distributed and reference table statements

Run from psql.

\COPY (SELECT 'SELECT create\_distributed\_table(''' || logicalrelid::text || ''',' ||'''' || column\_to\_column\_name(logicalrelid,partkey)||''');' from pg\_dist\_partition where partmethod='h') to distributed.sql;

\COPY (SELECT 'SELECT create\_reference\_table(''' || logicalrelid::text || ''');' from pg\_dist\_partition where partmethod='n') to reference.sql;

## Data-only Dump

Run from bash/terminal.

pg\_dump -Fd --data-only --no-owner -j 8 -f dumpdir --schema=public "host=<hostname> port=5432 dbname=citus user=citus password=<password> sslmode=require"

Run the above command so that dumpdir uses the 2TB disk that you created.

We are using parallel dump using -j flag. This would run 8 threads in parallel to pull out different tables at the same time. This would speed up the dump.

The above command might take significant time to complete, so use screen or tmux or nohup to avoid losing the session due to timeouts, etc. Feel free to get some coffee! ☺

At the end of the above step you will have five artifacts:

1. schema.sql – Table definitions
2. indexes.sql – Index/Primary/Unique/Foreign key definitions
3. distributed.sql – create\_distributed\_table statements
4. reference.sql – create\_reference\_table statements
5. dumpdir – directory that stores the data-only dump.

# Steps on Destination Cluster

## Apply schema

Connect to the destination cluster via psql and run below commands.

\i schema.sql

## Distribute and make table referenced

Connect to the destination cluster via psql and run below commands.

\i distributed.sql

\i reference.sql

## Data-only restore

Run from bash/terminal.

pg\_restore -F d --host=<destination\_host\_name> --dbname=citus --username=citus -n public -j 4 dumpdir

The above command might take significant time to complete, so use screen or tmux or nohup to avoid losing the session due to timeouts, etc.

## Apply primary/unique/foreign keys and indexes

Connect to the destination cluster via psql and run below commands.

\i indexes.sql

The above command might take significant time to complete, so use screen or tmux or nohup to avoid losing the session due to timeouts, etc.

## Vacuum the database

Connect to the destination cluster via psql and run below commands.

VACUUM ANALYZE;

SELECT run\_command\_on\_workers($cmd$ VACUUM ANALYZE;$cmd$);

## Data validation

Run counts on all tables and make sure counts are same across both the clusters.