Guide: In-Place Major Version Upgrade & Backup of PostgreSQL on GCP

1. Plan a major version upgrade

1. Assess current and target versions:

To check the database versions that you can target for an in-place upgrade on your instance, do the following:

```
gcloud sql instances describe INSTANCE_NAME
```

- Replace INSTANCE_NAME with the name of your Cloud SQL instance. Look for the upgradableDatabaseVersions section in the output, which shows the versions you can target.
- Review Compatibility: Consult the GCP Database Version Policy and your target version's release notes. Be aware that major upgrades may introduce incompatible changes that might require updates to application code, schema, or settings.

2. Review Changes and Incompatibilities

2. Prepare for a major version upgrade

1. Check and Adjust Character Set Configuration (LC_COLLATE)

Verify LC_COLLATE: Connect to the PostgreSQL instance and check LC_COLLATE:

```
SELECT datname, datcollate FROM pg_database WHERE datname IN ('template0', 'template1', 'postgres');
```

Update LC_COLLATE if LC_COLLATE is not en_US.UTF8 - complete the following steps to reconfigure:

- Dump the Database:

```
pg_dump -U USERNAME -h INSTANCE_IP -d DATABASE_NAME > db_backup.sql
```

- Drop the Database:

```
DROP DATABASE DATABASE_NAME;
```

- Create a New Database with en_US.UTF8 Encoding:

```
CREATE DATABASE DATABASE_NAME WITH ENCODING 'UTF8' LC_COLLATE = 'en_US.UTF8';
```

- Reload Data:

```
psql -U USERNAME -h INSTANCE_IP -d DATABASE_NAME < db_backup.sql
```

Alternative Method: Instead of dumping and recreating the database, you can rename it like below:

- Close all connections to the database.
- Rename the database.

```
ALTER DATABASE DATABASE_NAME RENAME TO NEW_NAME;
```

- Update your application configurations to use the new database name.
- Create a new, empty database with the default encoding.

2. Review and Manage Read Replicas:

Objective: Disable or delete read replicas, as cross-version replication is not supported.

Disable or Delete Each Replica - do the following steps:

- **1**: List Read Replicas:

```
gcloud sql instances list --filter="masterInstanceName=PRIMARY_INSTANCE_NAME"
```

- **2**: Delete Each Read Replica:

```
gcloud sql instances delete READ_REPLICA_NAME
```

Replace READ_REPLICA_NAME with the name of the replica instance. You'll be prompted to confirm the deletion.

3. Disable pglogical Logical Replication Before Upgrade:

Objective: Ensure that all pglogical logical replication configurations are disabled to prevent replication conflicts during the PostgreSQL upgrade.

Steps:

1: Identify Existing Subscriptions:

- Connect to your primary PostgreSQL instance using a client (such as psql) and list all existing pglogical subscriptions with the following command:

```
SELECT * FROM pglogical.show_subscription_status();
```

Review the output to identify active subscriptions.

2: Disable Each Subscription:

- For each subscription, disable it by using the pglogical.alter_subscription_disable function.
- Replace subscription_name with the actual name of the subscription and set the immediate parameter to true if the subscription needs to be disabled right away.

```
SELECT * FROM pglogical.alter_subscription_disable('subscription_name', true);
```

This command will stop the subscription and disconnect it from the provider immediately.

3: Drop Replication Slots:

- Once the subscriptions are disabled, drop any associated replication slots to free up resources and avoid conflicts.
- Run the following command to remove all replication slots related to pglogical:

```
SELECT pg_drop_replication_slot(slot_name) FROM pg_replication_slots
  WHERE slot_name IN (SELECT slot_name FROM pg_replication_slots);
```

4: Confirm Status:

- Confirm that all pglogical subscriptions and replication slots have been disabled by re-running the initial commands:

```
SELECT * FROM pglogical.show_subscription_status();
```

4. Manage PostgreSQL Extensions Before Upgrade

Objective: Ensure that all PostgreSQL extensions are compatible with the target version to prevent issues during the upgrade.

Steps:

1: Identify and Review Existing Extensions

2: Remove Unsupported Extensions

3: Upgrade PostGIS Extensions (if applicable)

- PostGIS is supported for Cloud SQL for PostgreSQL across all major versions. Ensure that your PostGIS extension is compatible with the target PostgreSQL version you plan to upgrade to.
- Check the specific version of PostGIS installed on your instance. For the upgrade, ensure the version is supported for the target PostgreSQL version. Below are the PostGIS extension versions for different Cloud SQL PostgreSQL versions:

Cloud SQL for PostgreSQL Version	PostGIS Extension Version
PostgreSQL 9.6	3.2.5
PostgreSQL 10	3.2.5
PostgreSQL 11	3.2.5
PostgreSQL 12	3.2.5

Cloud SQL for PostgreSQL Version	PostGIS Extension Version
PostgreSQL 13	3.4.0
PostgreSQL 14	3.4.0
PostgreSQL 15	3.4.0
PostgreSQL 16	3.4.0
PostgreSQL 17	3.4.3

- To specify the version of PostGIS in your CREATE EXTENSION command, use the VERSION clause.
- For more information on PostGIS installation and management, refer to the https://postgis.net/

4: Remove Deprecated Database Objects

- After cleaning up deprecated database objects, run the following SQL command to confirm there are no warnings before proceeding with the upgrade:

```
SELECT PostGIS_full_version();
```

- If there are no warnings in the output, you can proceed with the upgrade.

5: Verify Permissions for Extension Management

- Only users with the cloudsqlsuperuser role (such as the default postgres user) can manage extensions. Ensure you are using a user with these permissions to manage extensions as needed.

5. Note upgrade limitations

- PostGIS Version – Before upgrading to PostgreSQL 16 or later, you must update the PostGIS extension to version 3.4.0. If you're using PostgreSQL versions 9.6, 10, 11, or 12, you need to first perform an intermediate upgrade to version 13, 14, or 15, as older versions of PostgreSQL do not support PostGIS 3.4.0.
- Incompatible Extensions – If the `pgRouting` or `pg_squeeze` extensions are installed, a major version upgrade is not possible. You must uninstall these extensions before proceeding with the upgrade.
- Incompatible Flags – The flags `vacuum_defer_cleanup_age` and `force_parallel_mode` block the upgrade process. These flags must be removed prior to the upgrade. Note that in PostgreSQL 16 and later, the `vacuum_defer_cleanup_age` flag is deprecated, and `force_parallel_mode` has been renamed to `debug_parallel_query`.

6. Check Database Connectivity and Compatibility

- When performing an upgrade from one major version to another, attempt to connect to each database to see if there are any compatibility issues. Ensure that your databases can connect to each other. Check the `datallowconn` field for each database to ensure that a connection is allowed. A t value means that it's allowed, and an f value indicates that a connection can't be established.

3. Upgrade the Database Major Version In-Place

1. Initiate the Upgrade:

- Use the gcloud Command: The upgrade process is initiated by running the gcloud sql instances patch command. Replace the placeholder values for your specific instance and the desired target database version:

```
gcloud sql instances patch INSTANCE_NAME --database-version=DATABASE_VERSION
```

- Instance Unavailability: The instance will become unavailable for a period while the upgrade is performed, so plan for downtime.

2. Monitor the Upgrade Process:

- Get Upgrade Operation Status: Use the following command to monitor the status of the upgrade operation:

```
gcloud sql operations list --instance=INSTANCE_NAME
```

- Track the Upgrade with gcloud: Once you have the operation name, you can track its progress using the describe command:

```
gcloud sql operations describe OPERATION
```

4. Automatic Pre-Upgrade and Post-Upgrade Backups

1. View a list of backups

Once you perform a major version upgrade, Cloud SQL automatically creates two on-demand backups as part of the upgrade process:

- Pre-Upgrade Backup: Created before the upgrade starts.

- Post-Upgrade Backup: Created immediately after the upgrade finishes and new writes are allowed.

You can list these backups using the gcloud CLI:

```
gcloud sql backups list --instance=INSTANCE_NAME
```

2. Filter by Backup Type

The backups related to the upgrade will be labeled with Pre-upgrade or Post-upgrade in their description. For example, if upgrading from PostgreSQL 14 to 15, they will be labeled as:

- Pre-upgrade backup, POSTGRES_14 to POSTGRES_15
- Post-upgrade backup, POSTGRES_14 to POSTGRES_15

You can use this information to quickly identify the backups relevant to the upgrade.

3. View Backup Details

To get detailed information about any of the backups (such as start and end times), use the following command with the backup ID:

```
gcloud sql backups describe BACKUP_ID --instance=INSTANCE_NAME
```

This will allow you to confirm the exact time the backup was taken, which can be useful for verification or recovery purposes.

5. Complete the Major Version Upgrade

1. Re-enable pglogical Replication (if used)

Steps to re-enable pglogical replication:

- Drop the Existing Subscription on the Replica: You can drop the existing subscription on the replica using the following SQL command:

```
SELECT pglogical.drop_subscription(subscription_name := 'subscription_name');
```

Replace 'subscription_name' with the name of your existing subscription

- Recreate the Subscription on the Destination Replica: After dropping the old subscription, recreate it on the destination replica with the correct connection details to the primary instance:

```
SELECT pglogical.create_subscription(
    subscription_name := 'test_sub',
    provider_dsn := 'host=primary-ip port=5432 dbname=postgres user=replication_user password=replicapassword'
);
```

Replace 'test_sub' with your desired subscription name and use the actual IP address and replication credentials of the primary instance.

- Check the Subscription Status: Verify that the subscription is working properly by checking its status:

```
SELECT * FROM pglogical.show_subscription_status('test_sub');
```

2. Recreate Read Replicas (if deleted before upgrade)

If you deleted any read replicas before the upgrade, you can create new read replicas after upgrading the primary instance. These new replicas will automatically be provisioned with the latest database version. Steps to create a new read replica:

- **1**: Ensure the Primary Instance is Upgraded.
- **2**: Create a New Read Replica: You can use the gcloud sql instances create command to create a new read replica. The command is as follows:

```
gcloud sql instances create REPLICA_INSTANCE_NAME \
--master-instance-name=PRIMARY_INSTANCE_NAME \
--database-version=VERSION
```