IBP PostgreSQL Migration Tool User Guide

Introduction

The IBP PostgreSQL Migration Tool is a powerful Bash script designed to automate the migration of an IBP (Integrated Business Planning) EC2 instance to a new environment. It handles the entire end-to-end process, from backing up databases and server configuration files on a source server to restoring them on a new destination server. The script is executed from a jumpbox that has SSH access to both the source and destination machines.

This migration process is ideal for upgrading the underlying infrastructure, such as moving to a newer operating system (e.g., Ubuntu 22.04) and a more recent version of PostgreSQL (e.g., PostgreSQL 14), without requiring manual intervention for each step.

How it Works

The migration tool performs a series of automated steps to ensure a safe and reliable migration. The high-level workflow is as follows:

1. **Pre-flight Checks**: Before starting the migration, the script validates the environment. It checks for SSH connectivity to both the source and destination servers and verifies that there is sufficient disk space for the backups.
2. **Backup**: The script connects to the source server and performs a full backup of all PostgreSQL databases. It also archives critical server configuration files, such as those for Jetty, SSH host keys, and other IBP-specific settings.
3. **Data Transfer**: The backups are compressed into a single archive file, which is then securely transferred from the source server to the destination server via the jumpbox.
4. **Restore**: On the destination server, the script restores the PostgreSQL databases and moves the server configuration files to their correct locations.
5. **Post-Restore Tasks**: After the restoration is complete, the script performs several maintenance tasks, such as running `ANALYZE`, `VACUUM`, and `REINDEX` on the databases to ensure optimal performance.

Prerequisites

Before running the migration tool, please ensure that the following requirements are met:

- You have a **jumpbox** with SSH access to both the source and destination servers. The script is intended to be run from this jumpbox.
- The SSH user on the source and destination servers must have **passwordless `sudo` privileges**. This is necessary for the script to perform tasks that require elevated permissions, such as creating directories and managing services.
- The source and destination servers must have **PostgreSQL installed and running**. The script does not handle the installation of PostgreSQL.
- The `rsync` and `zstd` packages must be installed on both the source and destination servers. The script will attempt to install `zstd` if it is not present, but it is recommended to have it pre-installed.

Usage

The script can be run in two ways: by providing the connection details via environment variables or by entering them interactively when prompted.

Using Environment Variables

You can set the following environment variables before running the script:

- `SOURCE_HOST`: The hostname or IP address of the source server.
- `SOURCE_SSH_USER`: The SSH username for the source server (defaults to `smoothie`).
- `SOURCE_PORT`: The PostgreSQL port on the source server (defaults to `27095`).
- `DEST_HOST`: The hostname or IP address of the destination server.
- `DEST_SSH_USER`: The SSH username for the destination server (defaults to `smoothie`).
- `DEST_PORT`: The PostgreSQL port on the destination server (defaults to `27095`).

Once the environment variables are set, you can run the script as follows:

```
./pg-migrator.sh
```

Interactive Mode

If you do not provide the connection details via environment variables, the script will prompt you to enter them interactively. Simply run the script without any environment variables set:

```
./pg-migrator.sh
```

The script will then ask for the source host, destination host, and other required information.

The Migration Process in Detail

The script executes a sequence of functions to complete the migration. Here is a breakdown of the main stages:

1. Initial Setup

The script begins by performing initial checks to ensure the environment is ready for migration. This includes:

- **Validating SSH Connections**: It tests the SSH connections to the source and destination servers to ensure they are accessible.
- **Checking Disk Space**: It verifies that there is at least 10GB of free disk space on both servers to accommodate the database dumps and other temporary files.
- **Creating Backup Directory**: It creates a backup directory on the source server, typically at `/tmp/pg_migration/dumps`.

2. Database Backup

Once the initial checks are complete, the script proceeds with backing up the databases on the source server. This involves:

- **Applying Maintenance Settings**: The script dynamically tunes the PostgreSQL configuration on the source server to optimize it for dump operations. This helps to speed up the backup process.
- **Exporting Global Objects**: It exports global objects from PostgreSQL, such as roles and tablespaces.
- **Dumping Databases**: It dumps all user databases in parallel to maximize efficiency.

The number of parallel jobs is calculated based on the available CPU cores.

3. Archiving and Transfer

After the databases are backed up, the script creates a compressed archive of the database dumps and essential server files. This archive is then transferred to the destination server.

- **Creating Archive**: A `.tar.zst` archive is created, containing the database dumps and server configuration files.
- **Transferring the Archive**: The archive is transferred from the source to the destination server through the jumpbox using `rsync`.

4. Database Restore

On the destination server, the script restores the databases and server files.

- **Applying Maintenance Settings**: Similar to the source server, the script tunes the PostgreSQL configuration on the destination server for optimal restore performance.
- **Restoring Global Objects**: It restores the global objects first.
- **Restoring Databases**: The script restores all databases in parallel. Before restoring, it drops any existing databases with the same name to ensure a clean restore.
- **Restoring Server Files**: It moves the server configuration files from the archive to their final locations on the destination server.

5. Post-Migration Validation

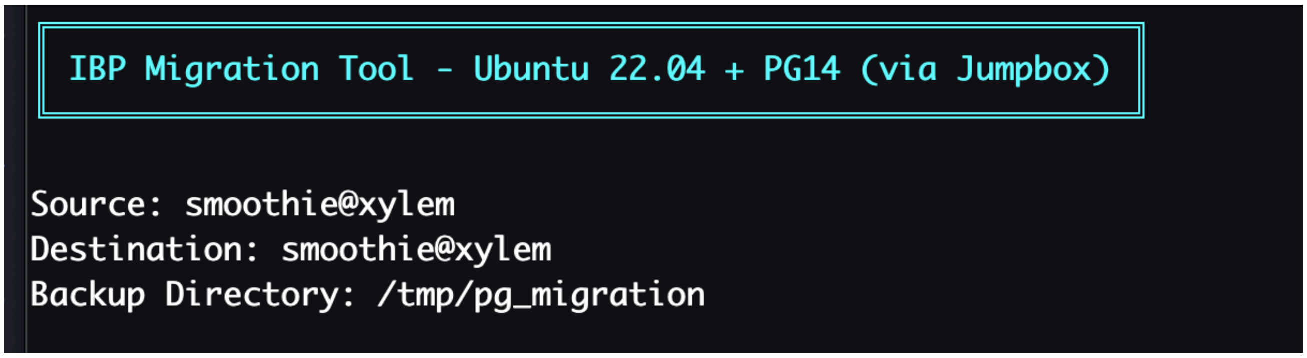
After the restore process is complete, the script performs several validation and maintenance tasks to ensure the new database is in a healthy state.

- **Running `ANALYZE`**: It runs the `ANALYZE` command to update statistics for the query planner.
- **Running `VACUUM`**: It performs a `VACUUM` to reclaim storage and improve performance.
- **Running `REINDEX`**: It rebuilds indexes to ensure they are optimal.
- **Validating Row Counts**: It checks the row counts of the tables to verify data integrity.

By following these steps, the IBP PostgreSQL Migration Tool provides a reliable and automated way to migrate your IBP instances with minimal downtime.

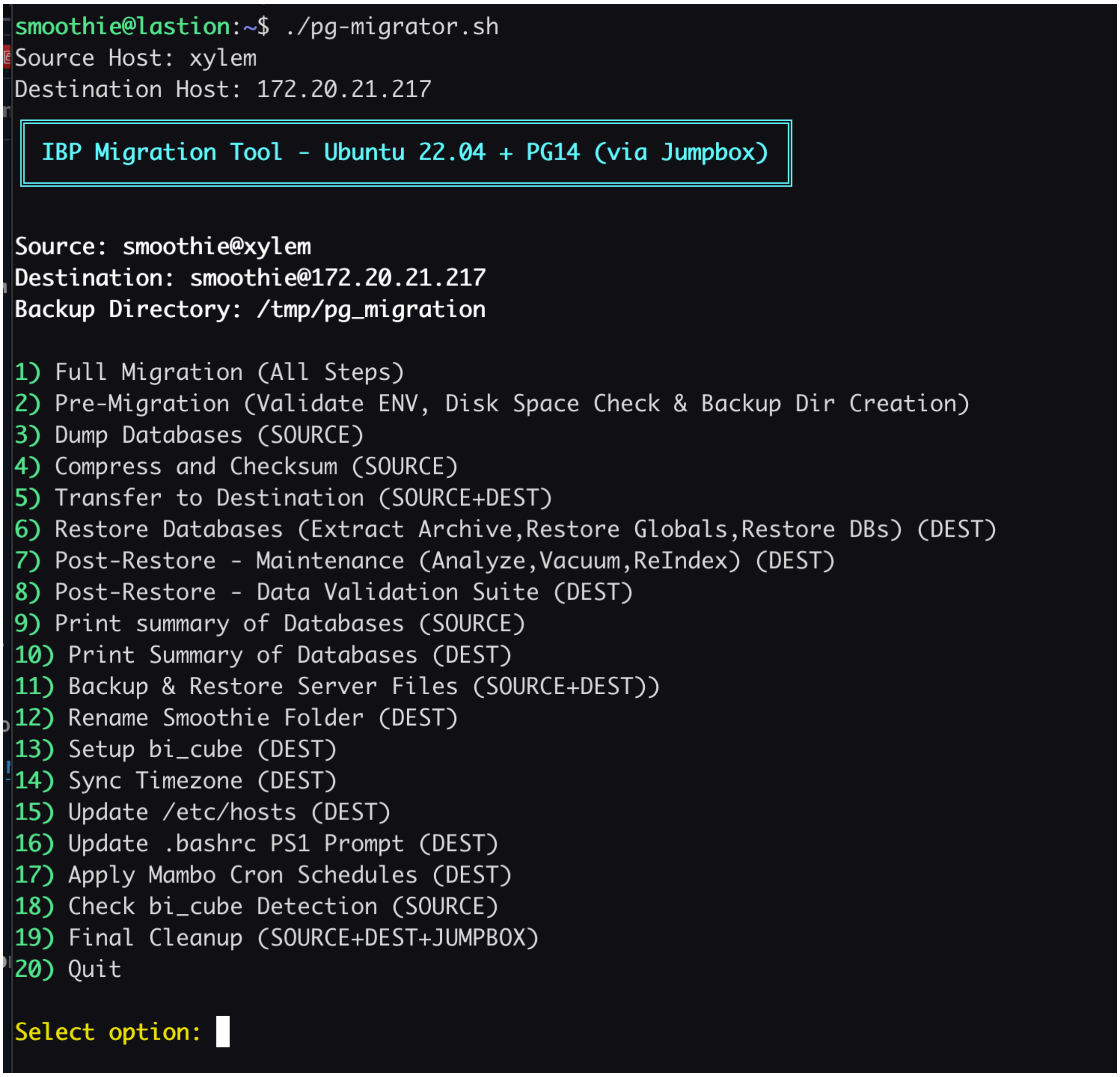
// … existing code …
The script will then ask for the source host, destination host, and other required information.

Once the script is running, you will see a header displaying the source and destination connection details.



```
IBP Migration Tool - Ubuntu 22.04 + PG14 (via Jumpbox)

Source: smoothie@xylem
Destination: smoothie@xylem
Backup Directory: /tmp/pg_migration
```

You will then be presented with a menu of options to choose from.



```
smoothie@lastion:~$ ./pg-migrator.sh
Source Host: xylem
Destination Host: 172.20.21.217

 IBP Migration Tool - Ubuntu 22.04 + PG14 (via Jumpbox)

Source: smoothie@xylem
Destination: smoothie@172.20.21.217
Backup Directory: /tmp/pg_migration

1) Full Migration (All Steps)
2) Pre-Migration (Validate ENV, Disk Space Check & Backup Dir Creation)
3) Dump Databases (SOURCE)
4) Compress and Checksum (SOURCE)
5) Transfer to Destination (SOURCE+DEST)
6) Restore Databases (Extract Archive,Restore Globals,Restore DBs) (DEST)
7) Post-Restore - Maintenance (Analyze,Vacuum,ReIndex) (DEST)
8) Post-Restore - Data Validation Suite (DEST)
9) Print summary of Databases (SOURCE)
10) Print Summary of Databases (DEST)
11) Backup & Restore Server Files (SOURCE+DEST))
12) Rename Smoothie Folder (DEST)
13) Setup bi_cube (DEST)
14) Sync Timezone (DEST)
15) Update /etc/hosts (DEST)
16) Update .bashrc PS1 Prompt (DEST)
17) Apply Mambo Cron Schedules (DEST)
18) Check bi_cube Detection (SOURCE)
19) Final Cleanup (SOURCE+DEST+JUMPBOX)
20) Quit

Select option:
```

The Migration Process in Detail

The script executes a sequence of functions to complete the migration. Here is a breakdown of the main stages:

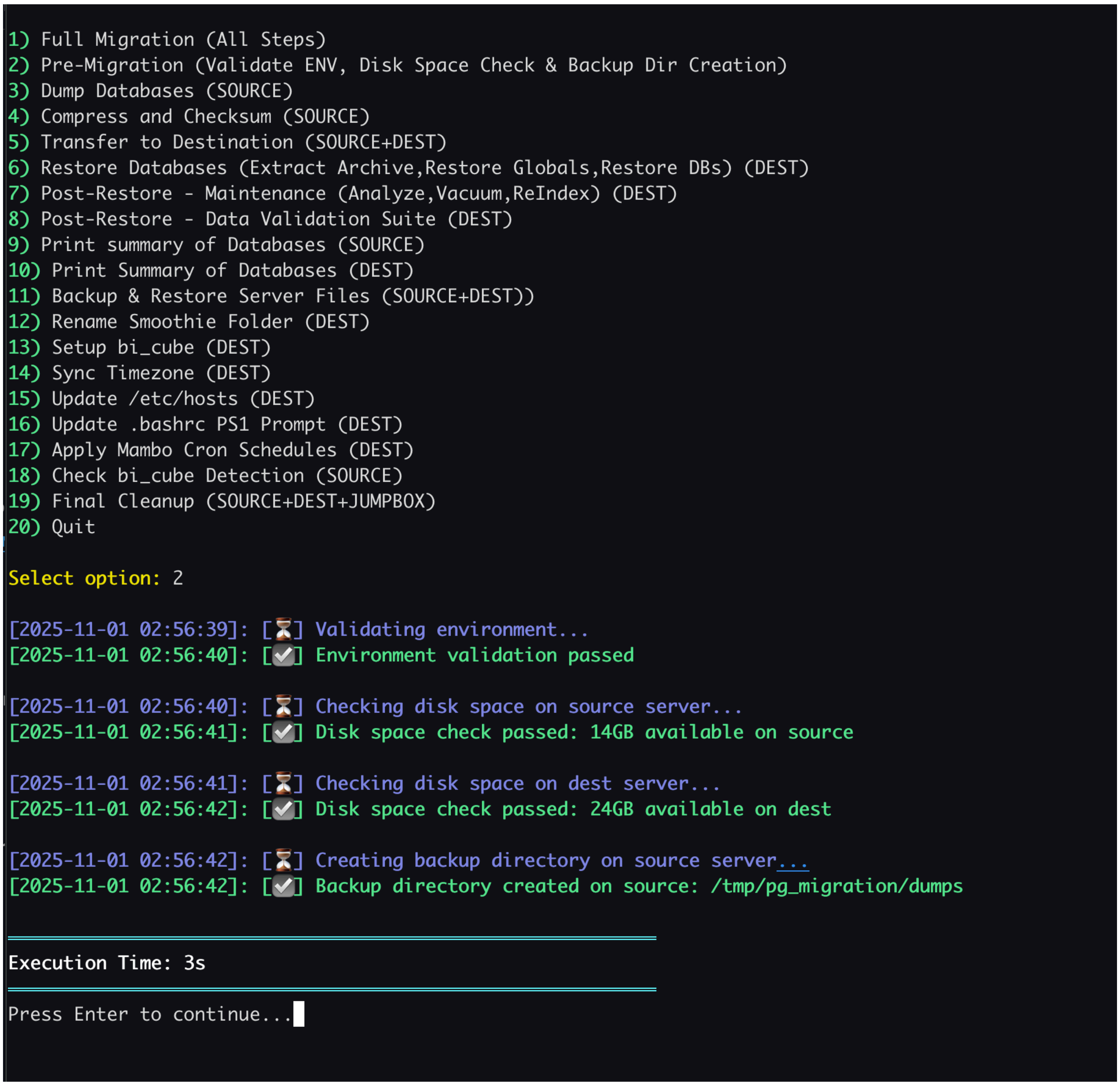
1. Initial Setup

The script begins by performing initial checks to ensure the environment is ready for migration. This includes:

- **Validating SSH Connections**: It tests the SSH connections to the source and destination servers to ensure they are accessible.
- **Checking Disk Space**: It verifies that there is at least 10GB of free disk space on both

servers to accommodate the database dumps and other temporary files.

- **Creating Backup Directory**: It creates a backup directory on the source server, typically at `/tmp/pg_migration/dumps`.



```
1) Full Migration (All Steps)
2) Pre-Migration (Validate ENV, Disk Space Check & Backup Dir Creation)
3) Dump Databases (SOURCE)
4) Compress and Checksum (SOURCE)
5) Transfer to Destination (SOURCE+DEST)
6) Restore Databases (Extract Archive,Restore Globals,Restore DBs) (DEST)
7) Post-Restore - Maintenance (Analyze,Vacuum,ReIndex) (DEST)
8) Post-Restore - Data Validation Suite (DEST)
9) Print summary of Databases (SOURCE)
10) Print Summary of Databases (DEST)
11) Backup & Restore Server Files (SOURCE+DEST))
12) Rename Smoothie Folder (DEST)
13) Setup bi_cube (DEST)
14) Sync Timezone (DEST)
15) Update /etc/hosts (DEST)
16) Update .bashrc PS1 Prompt (DEST)
17) Apply Mambo Cron Schedules (DEST)
18) Check bi_cube Detection (SOURCE)
19) Final Cleanup (SOURCE+DEST+JUMPBOX)
20) Quit

Select option: 2

[2025-11-01 02:56:39]: [⏳] Validating environment...
[2025-11-01 02:56:40]: [✅] Environment validation passed

[2025-11-01 02:56:40]: [⏳] Checking disk space on source server...
[2025-11-01 02:56:41]: [✅] Disk space check passed: 14GB available on source

[2025-11-01 02:56:41]: [⏳] Checking disk space on dest server...
[2025-11-01 02:56:42]: [✅] Disk space check passed: 24GB available on dest

[2025-11-01 02:56:42]: [⏳] Creating backup directory on source server...
[2025-11-01 02:56:42]: [✅] Backup directory created on source: /tmp/pg_migration/dumps

Execution Time: 3s

Press Enter to continue...
```

2. Database Backup

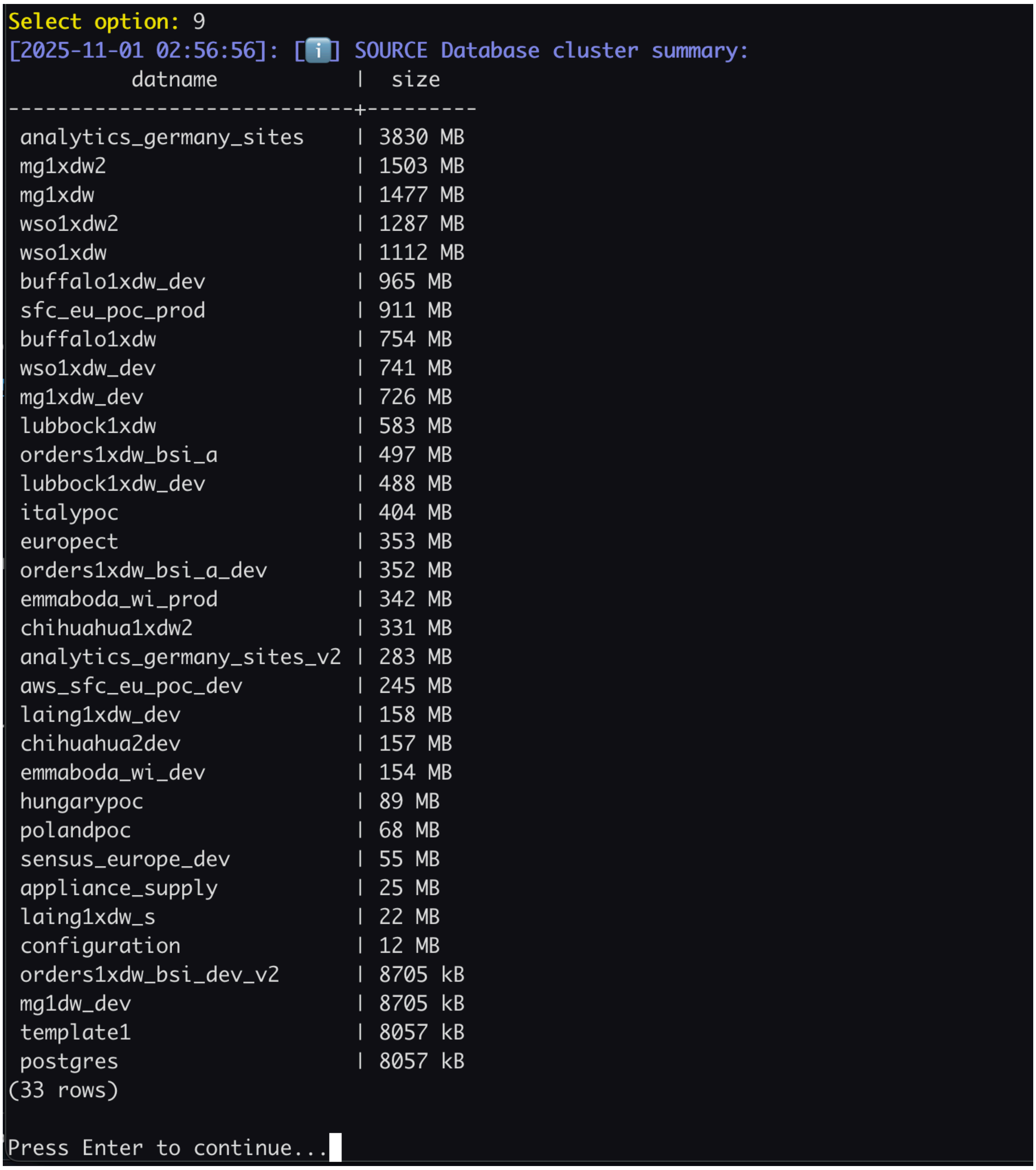
Once the initial checks are complete, the script proceeds with backing up the databases on the source server. This involves:

- **Applying Maintenance Settings**: The script dynamically tunes the PostgreSQL configuration on the source server to optimize it for dump operations. This helps to speed up the backup process.
- **Exporting Global Objects**: It exports global objects from PostgreSQL, such as roles

and tablespaces.

- **Dumping Databases**: It dumps all user databases in parallel to maximize efficiency. The number of parallel jobs is calculated based on the available CPU cores.

Before starting the backup, you can view a summary of the databases on the source server.



```
Select option: 9
[2025-11-01 02:56:56]: [ℹ️] SOURCE Database cluster summary:
          datname           |  size
----------------------------+---------
 analytics_germany_sites    | 3830 MB
 mg1xdw2                    | 1503 MB
 mg1xdw                     | 1477 MB
 wso1xdw2                   | 1287 MB
 wso1xdw                    | 1112 MB
 buffalo1xdw_dev            | 965 MB
 sfc_eu_poc_prod            | 911 MB
 buffalo1xdw                | 754 MB
 wso1xdw_dev                | 741 MB
 mg1xdw_dev                 | 726 MB
 lubbock1xdw                | 583 MB
 orders1xdw_bsi_a           | 497 MB
 lubbock1xdw_dev            | 488 MB
 italypoc                   | 404 MB
 europect                   | 353 MB
 orders1xdw_bsi_a_dev       | 352 MB
 emmaboda_wi_prod           | 342 MB
 chihuahua1xdw2             | 331 MB
 analytics_germany_sites_v2 | 283 MB
 aws_sfc_eu_poc_dev         | 245 MB
 laing1xdw_dev              | 158 MB
 chihuahua2dev              | 157 MB
 emmaboda_wi_dev            | 154 MB
 hungarypoc                 | 89 MB
 polandpoc                  | 68 MB
 sensus_europe_dev          | 55 MB
 appliance_supply           | 25 MB
 laing1xdw_s                | 22 MB
 configuration              | 12 MB
 orders1xdw_bsi_dev_v2      | 8705 kB
 mg1dw_dev                  | 8705 kB
 template1                  | 8057 kB
 postgres                   | 8057 kB
(33 rows)

Press Enter to continue...
```

3. Archiving and Transfer

After the databases are backed up, the script creates a compressed archive of the database dumps and essential server files. This archive is then transferred to the destination server.

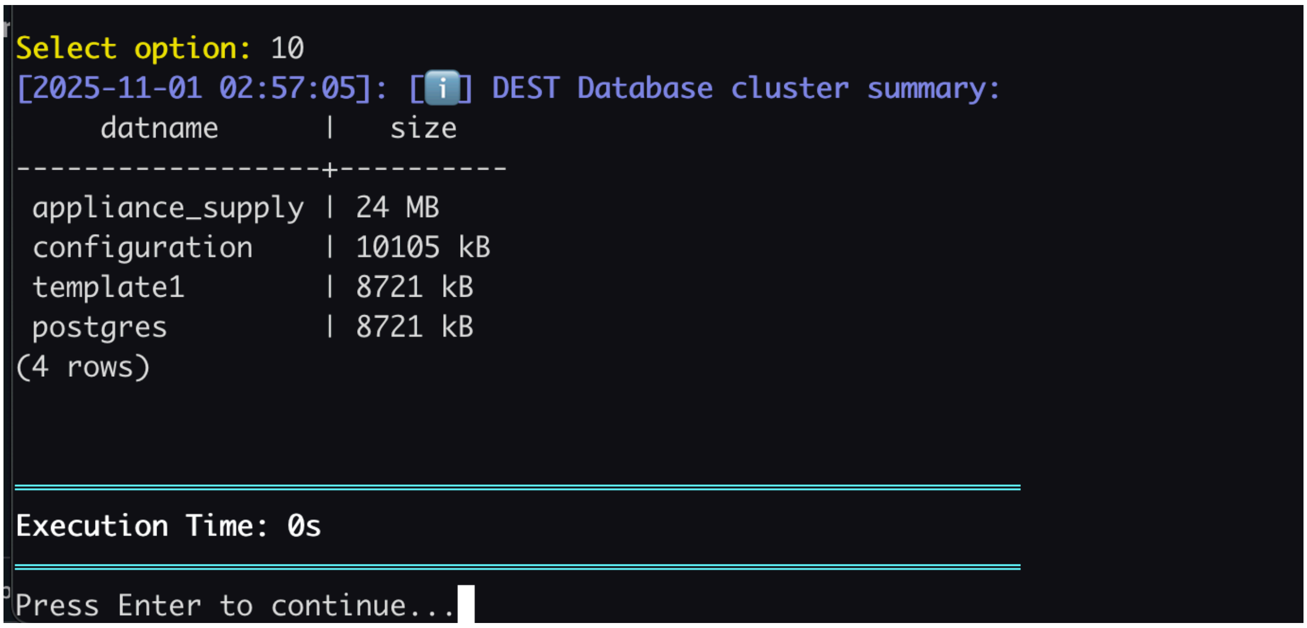
- **Creating Archive**: A `.tar.zst` archive is created, containing the database dumps and server configuration files.
- **Transferring the Archive**: The archive is transferred from the source to the destination server through the jumpbox using `rsync`.

4. Database Restore

On the destination server, the script restores the databases and server files.

- **Applying Maintenance Settings**: Similar to the source server, the script tunes the PostgreSQL configuration on the destination server for optimal restore performance.
- **Restoring Global Objects**: It restores the global objects first.
- **Restoring Databases**: The script restores all databases in parallel. Before restoring, it drops any existing databases with the same name to ensure a clean restore.
- **Restoring Server Files**: It moves the server configuration files from the archive to their final locations on the destination server.

After the restore, you can view a summary of the databases on the destination server.



```
Select option: 10
[2025-11-01 02:57:05]: [ℹ️] DEST Database cluster summary:
     datname       |   size
-------------------+----------
 appliance_supply | 24 MB
 configuration    | 10105 kB
 template1        | 8721 kB
 postgres         | 8721 kB
(4 rows)



Execution Time: 0s

Press Enter to continue...
```

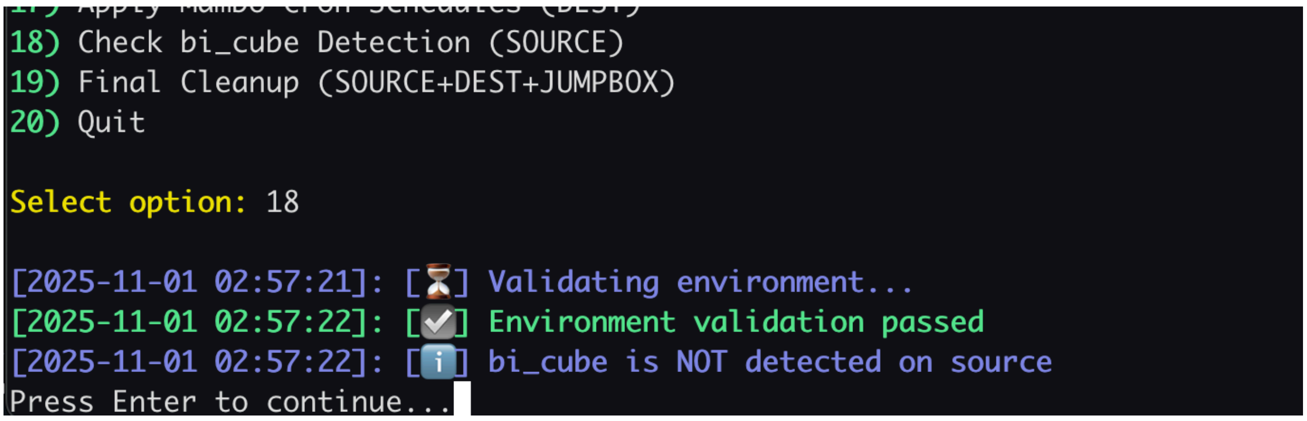
5. Post-Migration Validation

After the restore process is complete, the script performs several validation and maintenance tasks to ensure the new database is in a healthy state.

- **Running `ANALYZE`**: It runs the `ANALYZE` command to update statistics for the query planner.
- **Running `VACUUM`**: It performs a `VACUUM` to reclaim storage and improve performance.
- **Running `REINDEX`**: It rebuilds indexes to ensure they are optimal.
- **Validating Row Counts**: It checks the row counts of the tables to verify data integrity.

6. `bi_cube` Detection

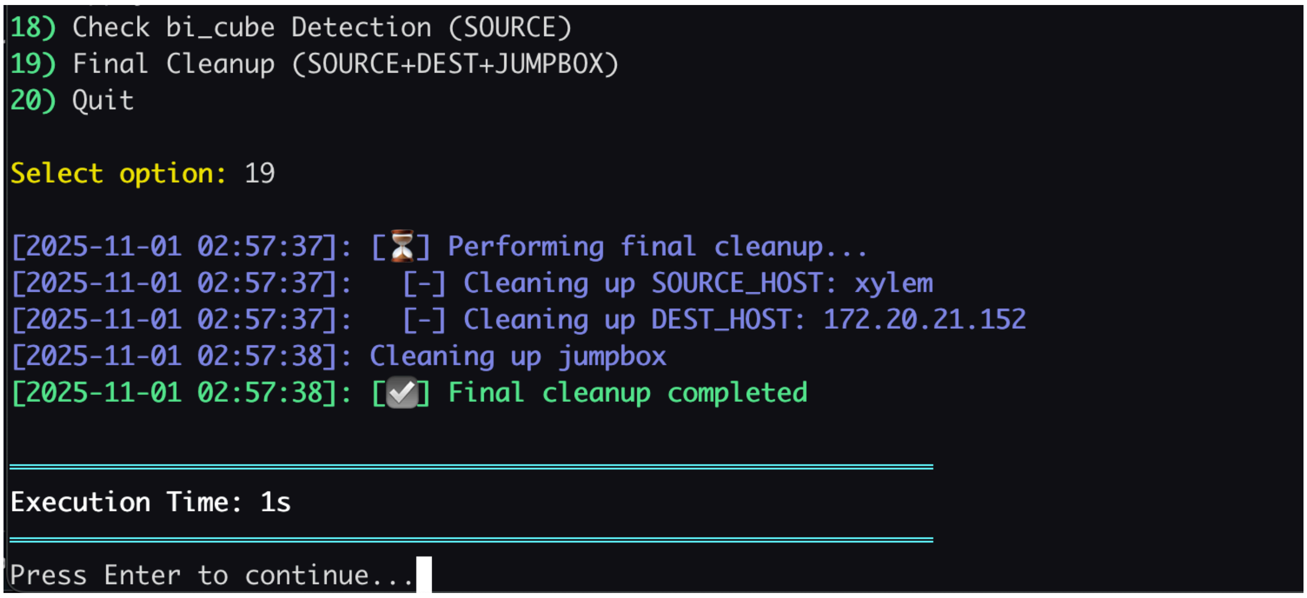
The script includes functionality to detect if the `bi_cube` is present on the source server. This is important for ensuring that all necessary files and configurations related to `bi_cube` are included in the migration.



```
17) Apply Mambo Cron Schedules (DEST)
18) Check bi_cube Detection (SOURCE)
19) Final Cleanup (SOURCE+DEST+JUMPBOX)
20) Quit

Select option: 18

[2025-11-01 02:57:21]: [⏳] Validating environment...
[2025-11-01 02:57:22]: [✅] Environment validation passed
[2025-11-01 02:57:22]: [ℹ️] bi_cube is NOT detected on source
Press Enter to continue...
```

7. Final Cleanup

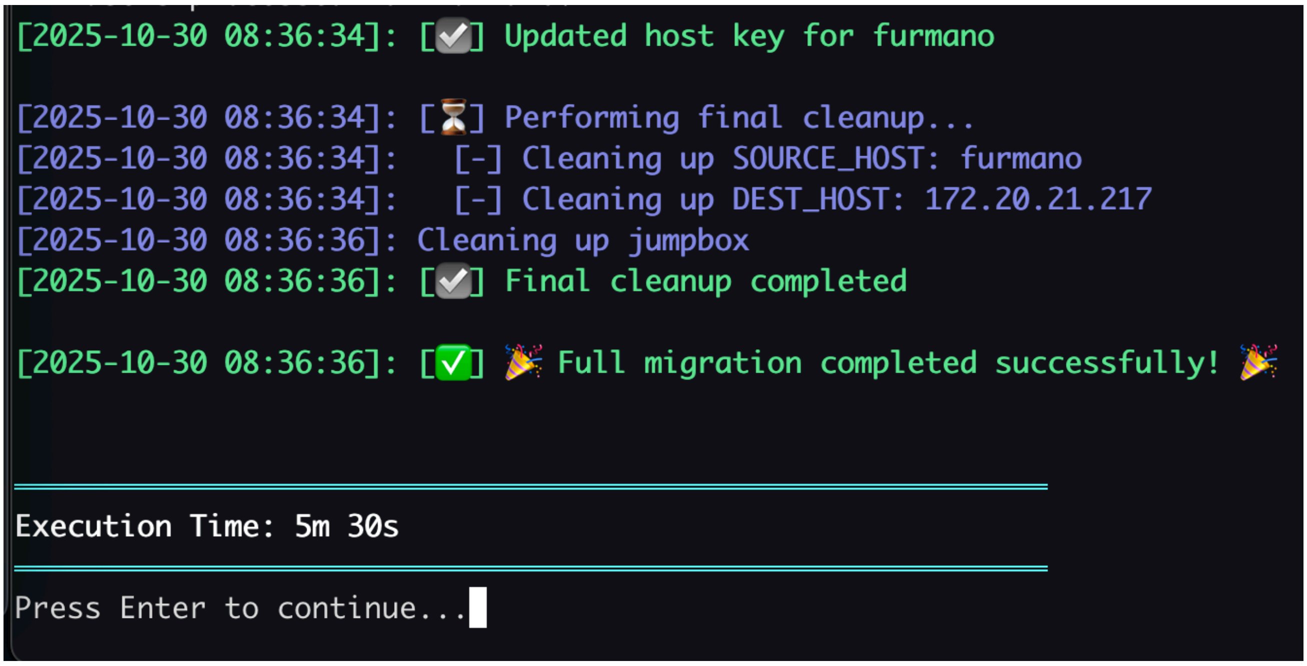
At the end of the migration, the script performs a cleanup process to remove temporary files from the source, destination, and jumpbox servers.



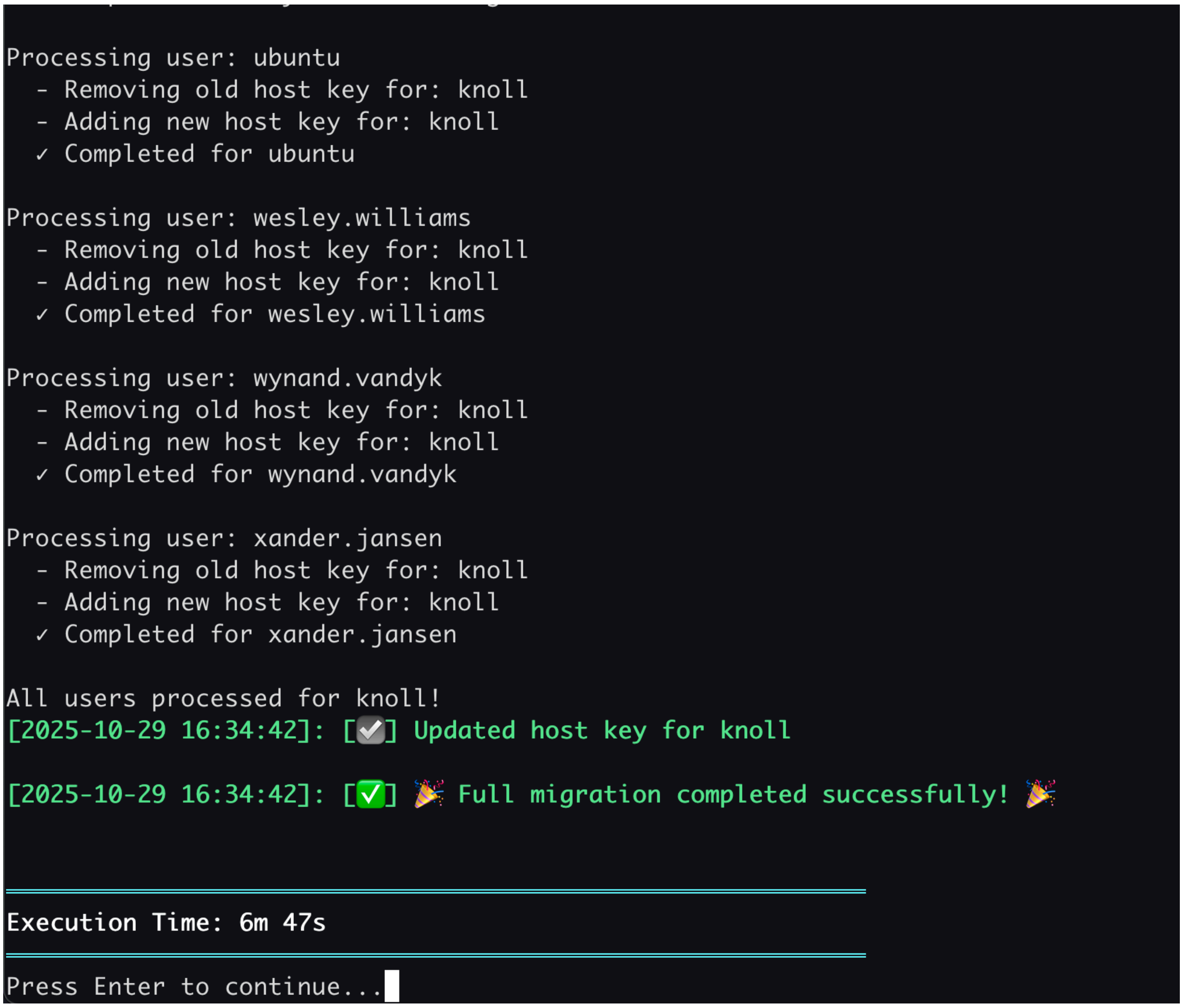
```
18) Check bi_cube Detection (SOURCE)
19) Final Cleanup (SOURCE+DEST+JUMPBOX)
20) Quit

Select option: 19

[2025-11-01 02:57:37]: [⏳] Performing final cleanup...
[2025-11-01 02:57:37]:   [-] Cleaning up SOURCE_HOST: xylem
[2025-11-01 02:57:37]:   [-] Cleaning up DEST_HOST: 172.20.21.152
[2025-11-01 02:57:38]: Cleaning up jumpbox
[2025-11-01 02:57:38]: [☑️] Final cleanup completed

════════════════════════════════════════════════════════════
Execution Time: 1s
════════════════════════════════════════════════════════════
Press Enter to continue...
```

Successful Migration

A successful migration will complete all the steps and display the total execution time.



```
[2025-10-30 08:36:34]: [☑️] Updated host key for furmano

[2025-10-30 08:36:34]: [⏳] Performing final cleanup...
[2025-10-30 08:36:34]:   [-] Cleaning up SOURCE_HOST: furmano
[2025-10-30 08:36:34]:   [-] Cleaning up DEST_HOST: 172.20.21.217
[2025-10-30 08:36:36]: Cleaning up jumpbox
[2025-10-30 08:36:36]: [☑️] Final cleanup completed

[2025-10-30 08:36:36]: [✅] 🎉 Full migration completed successfully! 🎉


════════════════════════════════════════════════════════════
Execution Time: 5m 30s
════════════════════════════════════════════════════════════
Press Enter to continue...
```



```
Processing user: ubuntu
  - Removing old host key for: knoll
  - Adding new host key for: knoll
  ✓ Completed for ubuntu

Processing user: wesley.williams
  - Removing old host key for: knoll
  - Adding new host key for: knoll
  ✓ Completed for wesley.williams

Processing user: wynand.vandyk
  - Removing old host key for: knoll
  - Adding new host key for: knoll
  ✓ Completed for wynand.vandyk

Processing user: xander.jansen
  - Removing old host key for: knoll
  - Adding new host key for: knoll
  ✓ Completed for xander.jansen

All users processed for knoll!
[2025-10-29 16:34:42]: [☑️] Updated host key for knoll

[2025-10-29 16:34:42]: [✅] 🎉 Full migration completed successfully! 🎉


════════════════════════════════════════════════════════════
Execution Time: 6m 47s
════════════════════════════════════════════════════════════
Press Enter to continue...
```

By following these steps, the IBP PostgreSQL Migration Tool provides a reliable and automated way to migrate your IBP instances with minimal downtime.