PostgreSQL DBA's Guide to Actifio Copy Data Management

Updated August 24, 2022

Contents

1 Introducing Actifio VDP for PostgreSQL Databases

Actifio Data Virtualization

An Actifio appliance is a highly scalable copy data management platform that virtualizes application data to improve the resiliency, agility, and cloud mobility of your business. It works by virtualizing data in much the same way other technologies have virtualized servers and networks. This enables you to capture data from production systems, manage it in the most efficient way possible, and use virtual copies of the data however they are needed.

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 30 years of active development on the core platform.

PostgreSQL backup API used by Actifio

Linux CBT and LVM snapshot: Postgres "pg_start_backup(<label>)" and "pg_stop_backup()" API with Linux CBT and LVM snapshot

File-based backups: PostgreSQL "pg_dump" File-based backups API

This provides the full backup of the database in backup format. Recovery API restore db will recover the database by physically overwriting the data area

PostgreSQL log backup: During a log backup we physically copy all the PostgreSQL WALs (WriteAheadLog). To purge logs, we use the OS level command.

```
(find $WALS_LOC -type f -mtime +$DELLOG -print -exec rm -f {} \;)
```

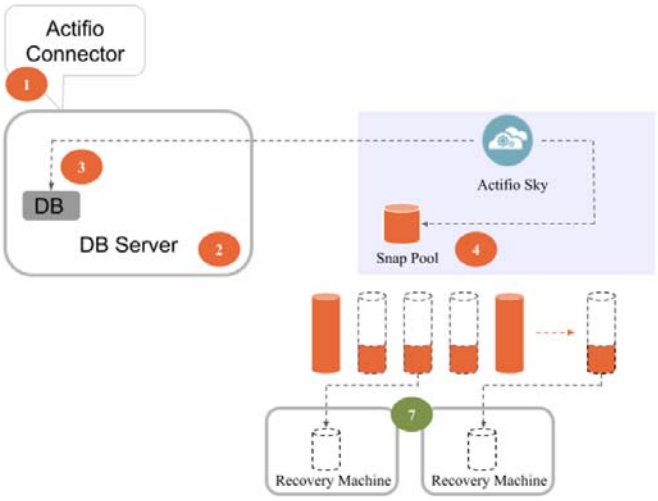
Data Capture

1. Actifio connector has CBT which keeps track of changed blocks in PostgreSQL database data area
2. Connector call PostgreSQL "pg_start_backup()" command before LVM snapshot
3. Connector creates LVM snapshot of PostgreSQL database data area and synthesize a bitmap
4. Connector call PostgreSQL "pg_stop_backup()" command and copies changed blocks
5. Connector deletes LVM snapshot and catalogs backup
6. Sky issues an internal snapshot and synthesize point-in-time virtual full

Data Recovery

7. For recovery, Actifio instantly mounts re-writable staging disk & brings DB online

How it Works: PostgreSQL with Linux CBT and LVM Snapshot



Data Capture

1. Actifio Connector is deployed in DB server
2. Mount staging disk on DB server
3. Invoke full backup using pg_dump backup command, writing backup to the mounted disk
4. Actifio takes an internal snapshot

Data Recovery

7. For recovery, Actifio instantly mounts the staging disk to DB server and kicks off a database restore

How it Works: PostgreSQL with File-Based Traditional Backup

2 Preparing the PostgreSQL Database

Prerequisites

- For best results, create a backup user with the 'superuser' and 'Replication' privileges.

  To create the backup user:

  ```
  create user actuser;
  alter user actuser with superuser;
  alter user actuser with Replication;
  ```

- If there are multiple PostgreSQL instances running on a server, then the backup username (and password if required) must be common for all PostgreSQL instances running on that server.
- PostgreSQL archive mode (archive_mode) must be set to ON to take log backup. To configure the archive log mode option, update the postgresql.conf file in the data director of the instance and reload the postgres.

  ```
  archive_mode = on
  pg_ctl reload -D <datadir>
  ```

Adding a PostgreSQL Database Host and Discovering the Instances and Databases

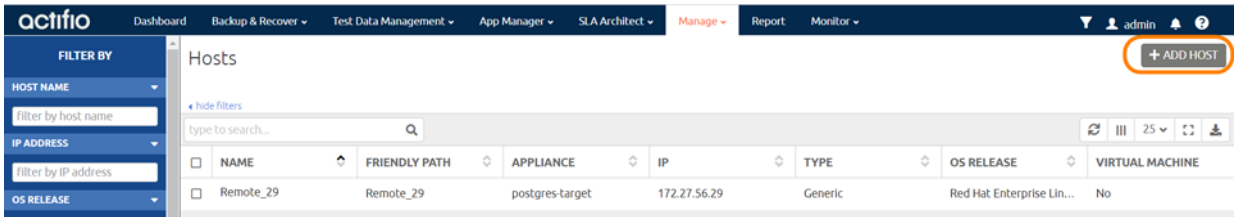
Before you can protect a PostgreSQL database, you must add the host and discover the database. This requires:

1. Adding the Host from Manage Hosts on page 3
2. Discovering the PostgreSQL Database Application from the Application Manager on page 6
3. Finding the Discovered PostgreSQL Instances and Databases in the Application Manager on page 6

Adding the Host from Manage Hosts

Add the host to AGM. If the host is already added then edit the host and make sure to set all the configurations correctly.

1. From the AGM Manage tab, select Hosts and click **+Add Host**.



2. On the Add Host page:

a. Name: Provide the PostgreSQL database server name.

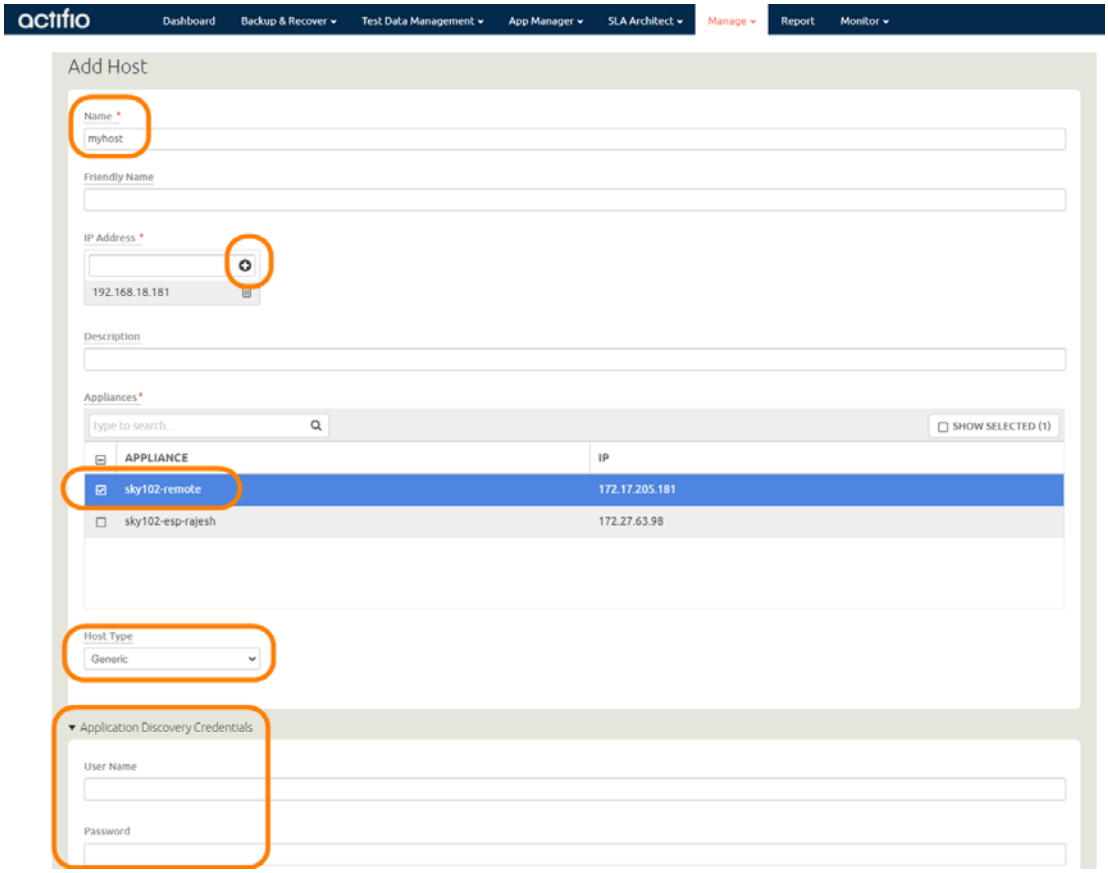
b. IP Address: Provide the PostgreSQL database server IP and click the **+** sign to the right.

c. Appliances: Select the check box for the appliance that will protect the host.

d. Host Type: Make sure this is **Generic**.

e. In Application Discovery Credentials, provide the credentials to discover the PostgreSQL instances and databases.

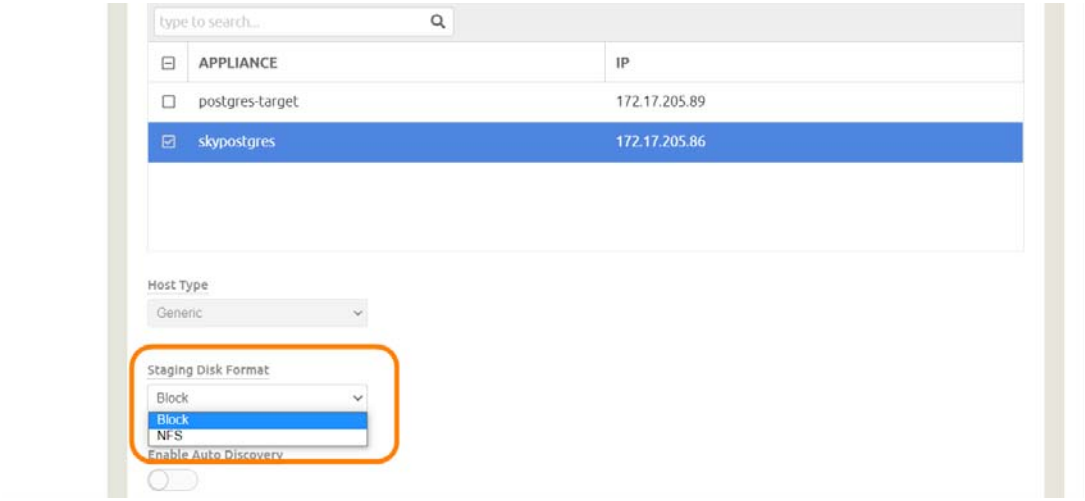
f. At the bottom of the page, click **Add** to add the host.



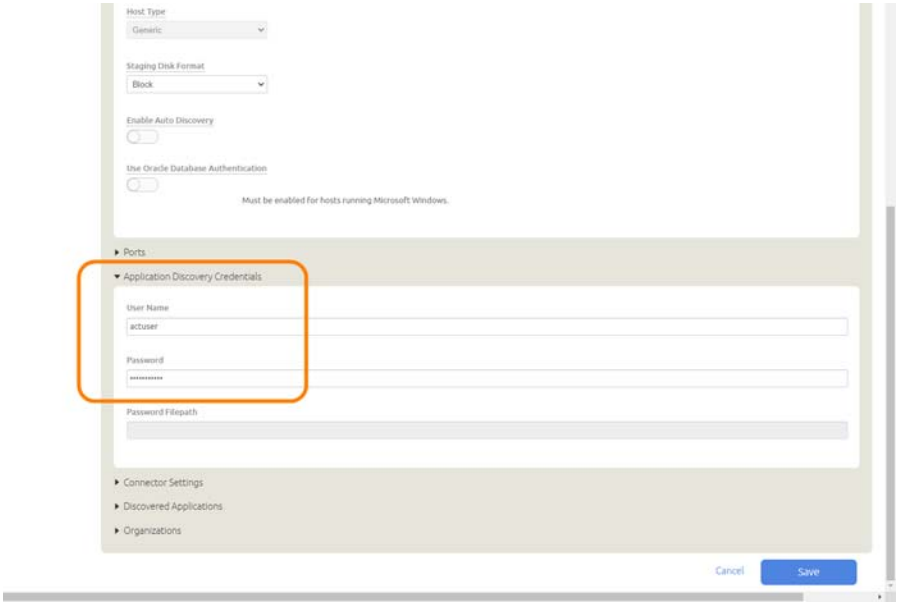
The Host will be added to the selected Actifio Appliance and managed from AGM.

3. On then next screen (Edit Host page), select the disk preference:

   - For block-based backup with CBT, select **Block**
   - For file-based backup with Full+Incremental file system backup: select **Block** *or* **NFS**



4. In Application Discovery Credentials, enter the username/password that you set up in Prerequisites on page 3.

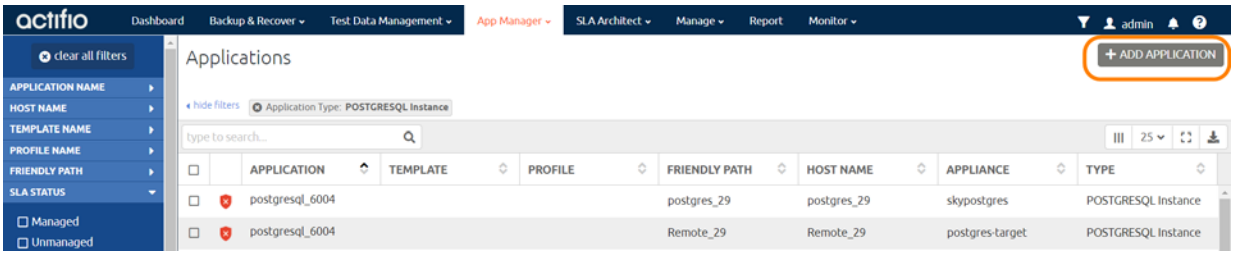


5. Click **Save** at the bottom of the page.

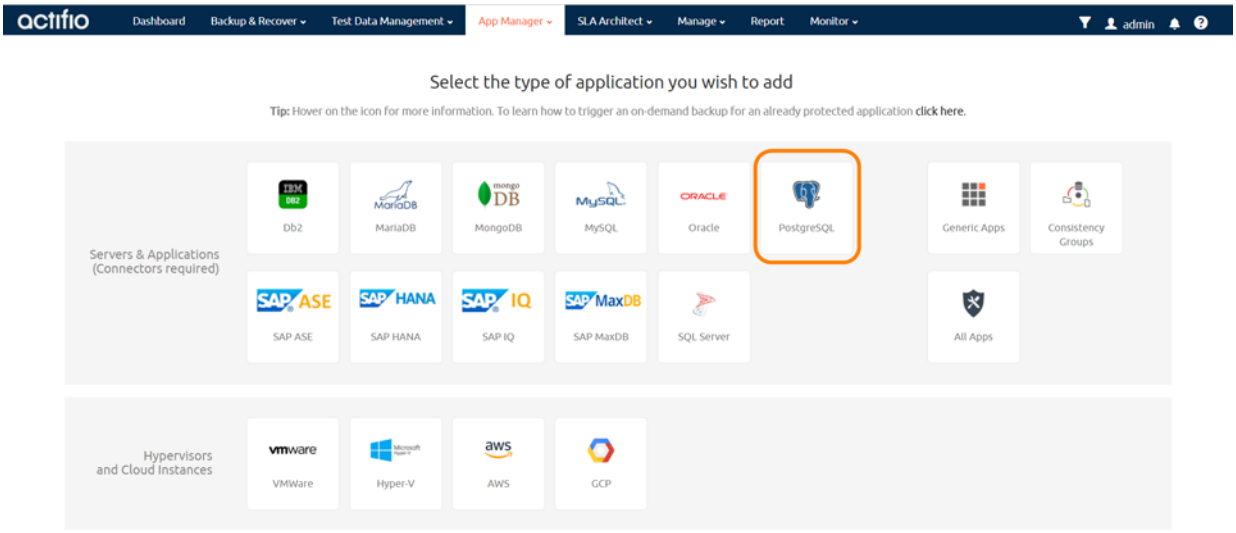
Discovering the PostgreSQL Database Application from the Application Manager

To discover PostgreSQL instances and databases:

1. From the AGM App Manager, Applications tab, select **Add Application** in the upper right corner.



2. On the Add Application page, select PostgreSQL, then follow the wizard.

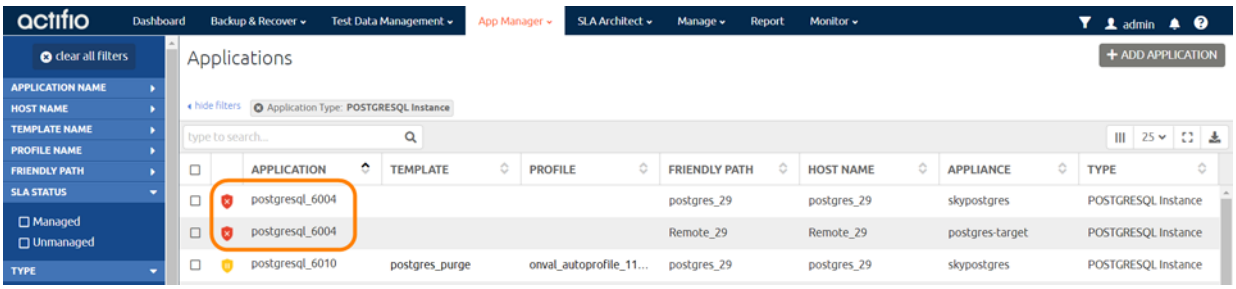


This will run the discovery on the PostgreSQL database host and will discover all PostgreSQL databases running on it.

Finding the Discovered PostgreSQL Instances and Databases in the Application Manager

To find the newly-discovered instances and databases, go to the AGM App Manager Applications tab. All applications known to the AGM of all types are listed. Use the Type application filter on the left pane to show only PostgreSQL databases and instances.

The new PostgreSQL databases will appear in the list as unmanaged (the red shield icon).



3 Configuring the Backup Method for a PostgreSQL Instance

After the instance is prepared and discovered as explained in Chapter 2, Preparing the PostgreSQL Database, you can configure a VDP backup method SLA for it. The procedures for developing SLAs are detailed in the AGM online help. This chapter provides information relevant to the PostgreSQL DBA.

You can capture PostgreSQL instances, with database inclusion and protection rules. After you capture the instance, you can mount individual databases and capture them for management.

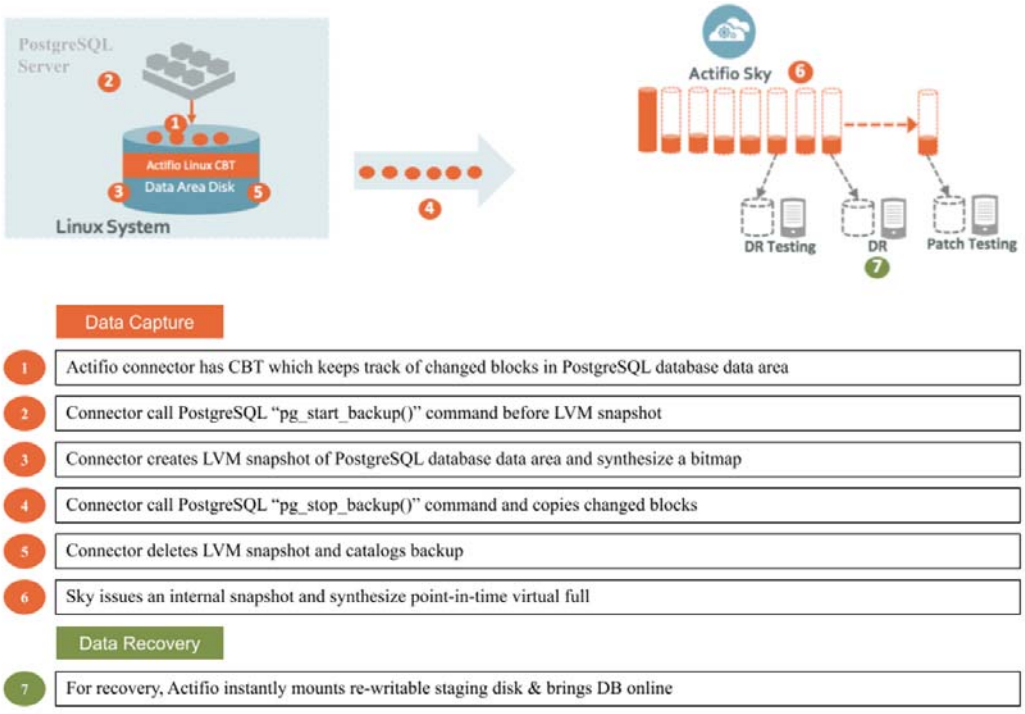
You can back up an instance:

- Using Block-Based LVM Snapshots with CBT on Linux on page 7
- Using File-Based Traditional Backups on page 8

Protection is set for the entire PostgreSQL Instance. You can include/exclude specific databases during the process using a Database Inclusion Rule from the Manage SLA page.

Using Block-Based LVM Snapshots with CBT on Linux

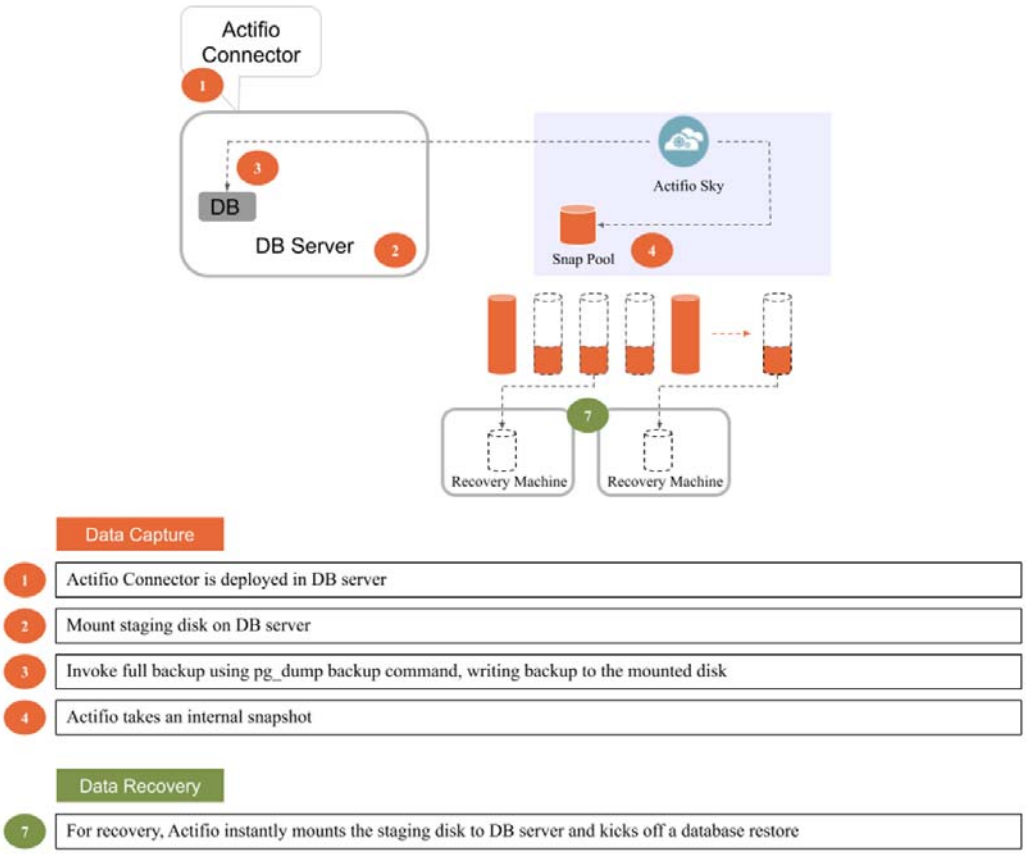
This takes advantage of Actifio changed-block tracking through the Actifio Connector for faster backups and recoveries. Actifio takes one full backup, and then backs up only the changed data blocks afterward.



How it Works: PostgreSQL with Linux CBT and LVM Snapshot

Using File-Based Traditional Backups

These are full (dump based) backups. They are larger and thus take more time and storage than the volume-based backups with CBT.



How it Works: PostgreSQL with File-Based Traditional Backup

Whichever method you select involves these steps:

Configuring Application Details & Settings on page 9

Ensuring that the Backup Capture Method is Set Correctly on page 11

Ensuring that the Staging Disk Format on the Host is Set Correctly on page 12

Configuring Application Details & Settings

Table 1: Application Details & Settings

Setting	Block-Based LVM Snapshot with CBT on Linux	File-Based Backup and Recovery, Block *or* NFS
Use Staging Disk Granularity as Minimum Staging Disk Size	For applications that are under the size of granularity setting that tend to periodically grow this new option is useful to avoid frequent costly FULL backups. Because the staging disk is thin provisioned, there is no initial cost to use a staging disk that is larger than required for immediate use. The values are 0 for No and the Staging Disk Granularity setting for Yes.	
Staging Disk Granularity	Maximum size of each staging disk when multiple staging disks are used for an application. The default value is 1000GB.	
Last Staging Disk Minimum Size	Minimum size of the last staging disk created for an application with multiple staging disks. This value is also used for additional disks allocated to accommodate growth. The default value is 250GB.	

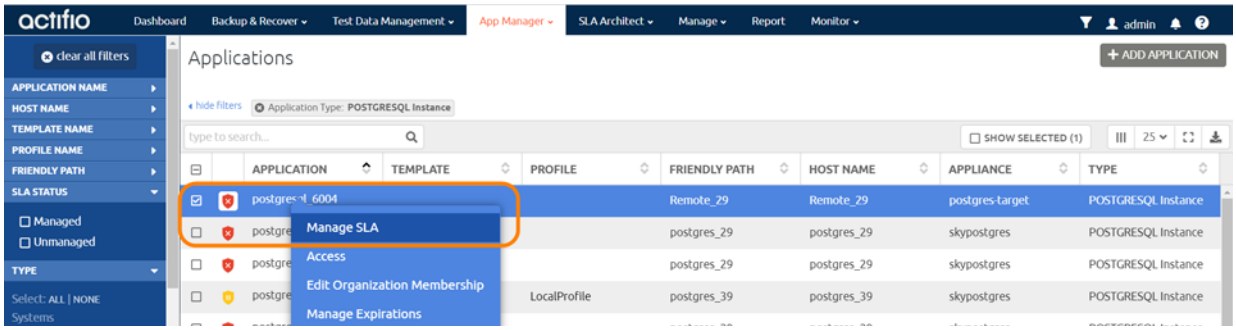
Table 1: Application Details & Settings

Setting	Block-Based LVM Snapshot with CBT on Linux	File-Based Backup and Recovery, Block *or* NFS
Connector Options	Use this only under the direction of Actifio Support.	
Percentage of Reserve Space in Volume Group	20% is recommended for LVM snapshot temporary space. Not applicable for protecting virtual databases.	Not applicable
Backup Capture Method	Use volume level backup	Use full+incremental filesystem backup
Force Full Filesystem Backup	Not applicable (PostgreSQL always takes a full backup)	
Database Filesystem Staging Disk Size in GB	Not applicable	Size of the database dump staging disk in GB. The default value is 100GB.
Log Backup Staging Disk Size in GB	The default value is ((daily log generation * retention of log backup SLA) plus 20% buffer). Default is recommended. Providing a value overrides the default. This becomes a fixed size and the log disk will no longer grow dynamically.	Not applicable
Retention of Production DB Logs in Days	This value is used to purge the log backup from basepath_logbackup destination. Based on this setting the last data backup id will be selected (CURRENT_TIMESTAMP, - the # days set) and the log will be purged older than the data backup id. Default value is 0 days. With default value all logs prior to last data backup will be purged.	
Script Timeout	The timeout value is applied to internal backup and recovery scripts called by connector. The default value is recommended.	

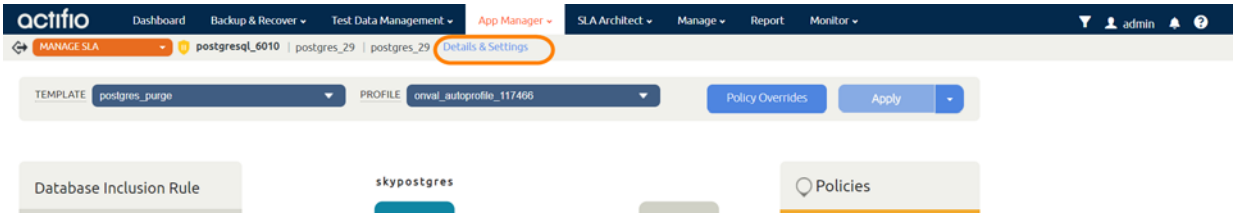
Ensuring that the Backup Capture Method is Set Correctly

Backup capture settings depend upon the backup capture method that you need. It is important to be certain that you have set the right backup method for your needs:

1. In the App Manager, Applications list, right-click the database and select **Manage SLA**.



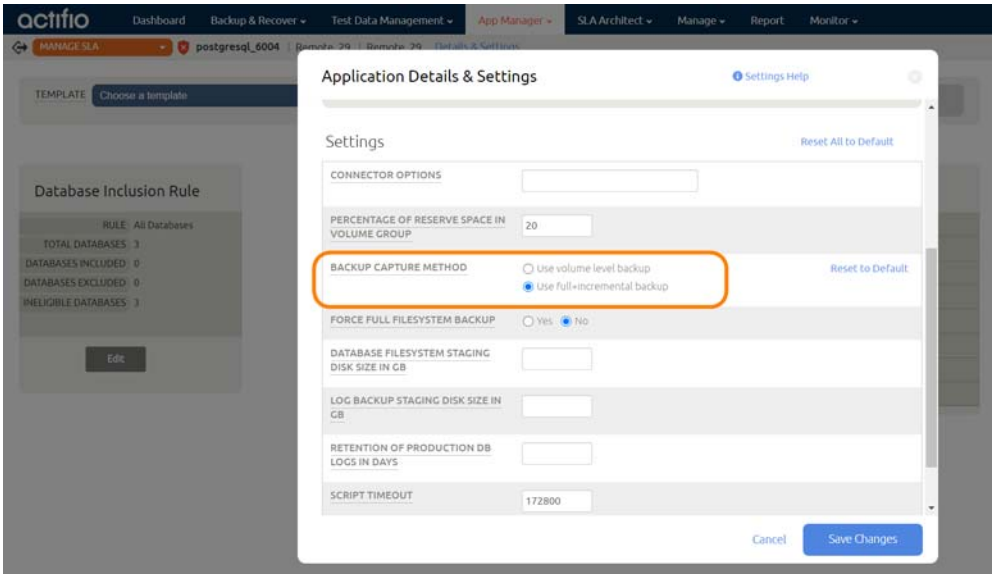
2. At the top of the Manage SLA page, select the **Details & Settings** link:



This opens the details and settings for this database. Check the Backup Capture Method:

- LVM Snapshot with Change Block Tracking: **Use volume level backup**
- Traditional Backup and Recovery API "file-based" backups: **Use full+incremental backup**

Note: *System databases on a root partition can be backed up as LVM Snapshots and later mounted as virtual databases, but they cannot be used in a traditional Restore operation, as the root partition cannot be unmounted. This will need manual restore and recovery from a simple mount back to the same host.*

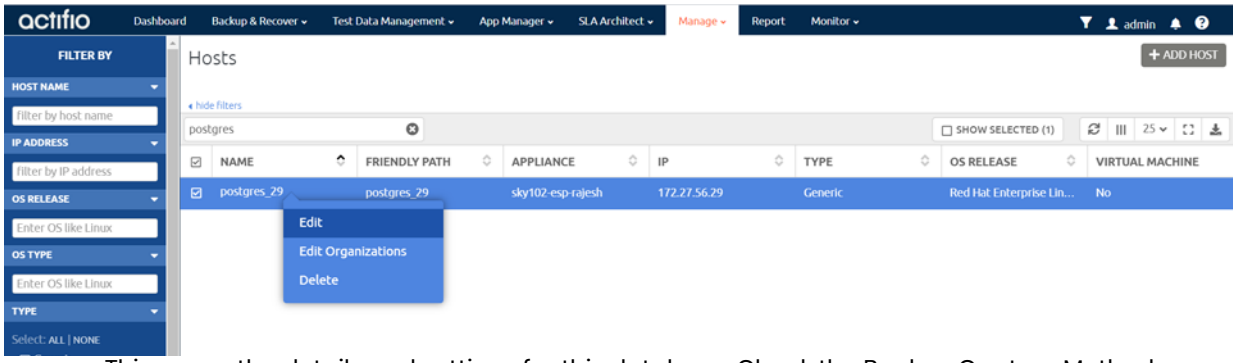


3. Click **Save** at the bottom of the page if you had to change anything.

Ensuring that the Staging Disk Format on the Host is Set Correctly

To set staging disk format for storage snapshots:

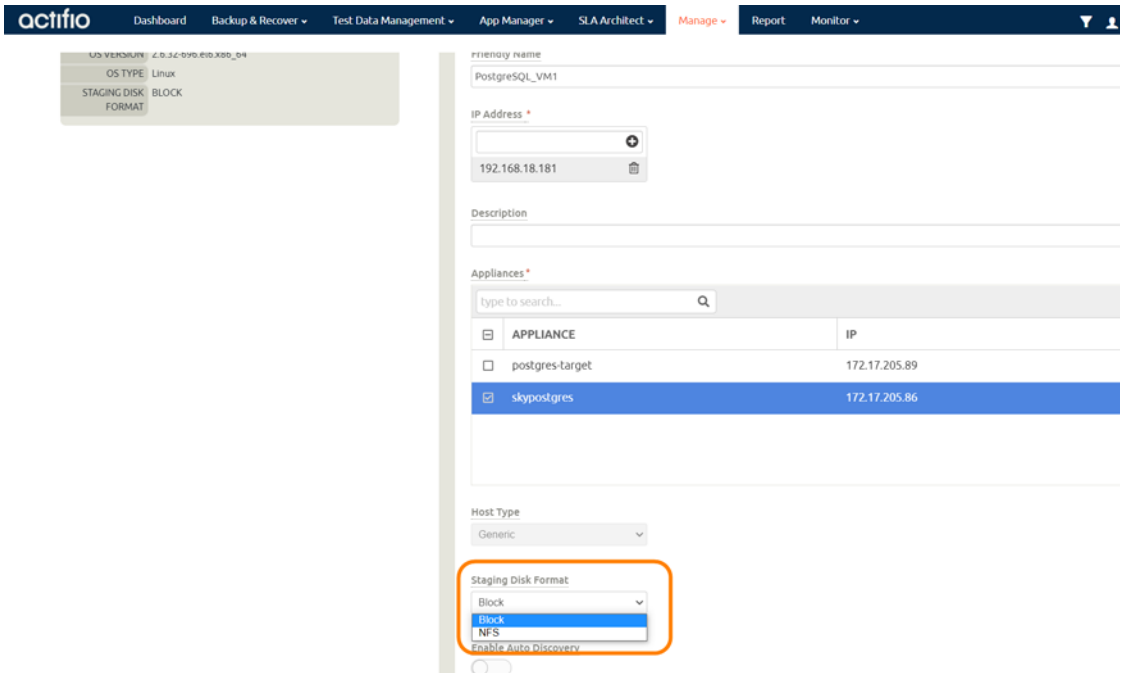
1. From the Manage, Hosts list, right-click the host and select **Edit**.



This opens the details and settings for this database. Check the Backup Capture Method:

- LVM Snapshot with Change Block Tracking: **Block**
- Traditional Backup and Recovery API "file-based" backups: **NFS** *or* **Block**

2. Set Staging Disk Format to **Block**.



3. Then click **Save** at the bottom of the page.

4 Protecting a PostgreSQL Instance and its Databases and Logs

After the SLA is configured as detailed in Chapter 3, Configuring the Backup Method for a PostgreSQL Instance, you can configure a VDP backup method for the database.

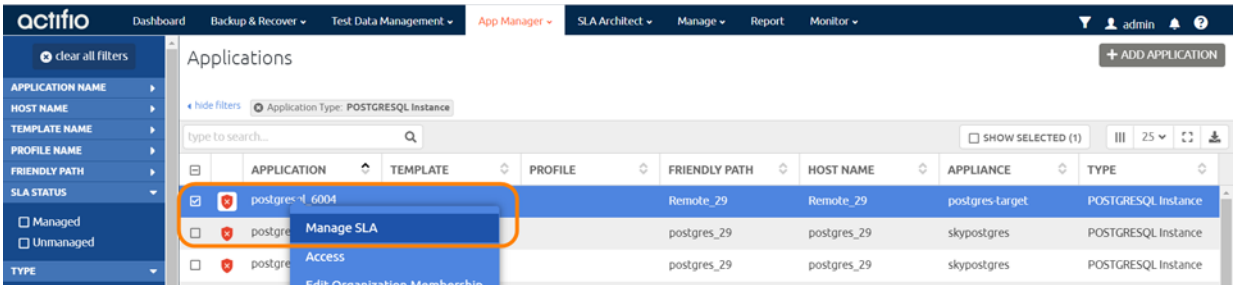
Protection is set for the entire PostgreSQL Instance. You can include/exclude specific databases during the process using a Database Inclusion Rule from the Manage SLA page.

This chapter includes:

Protecting a PostgreSQL Instance

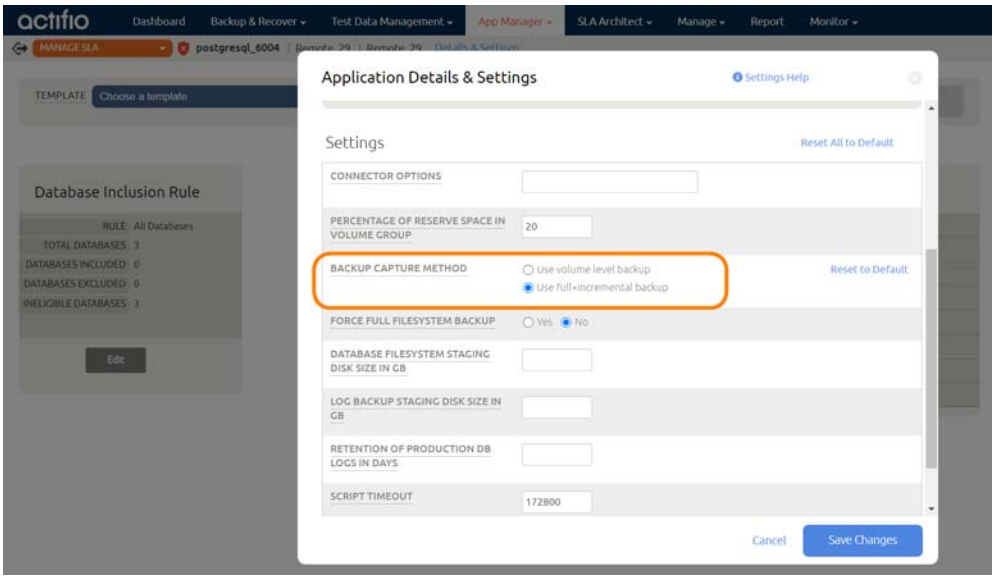
To protect the instance:

1. From the App Manager, Applications list, right-click the instance and select **Manage SLA**.



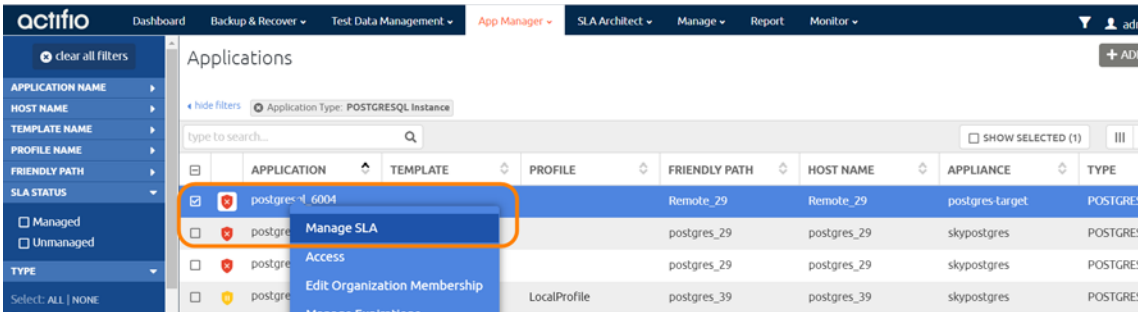
2. On the Manage SLA page, select a template and a resource profile, then click **Apply**.

3. On the Apply SLA page, make sure that the backup capture method matches the type of backup set in Chapter 3, Configuring the Backup Method for a PostgreSQL Instance. Click **Apply SLA** or **Save Changes**.

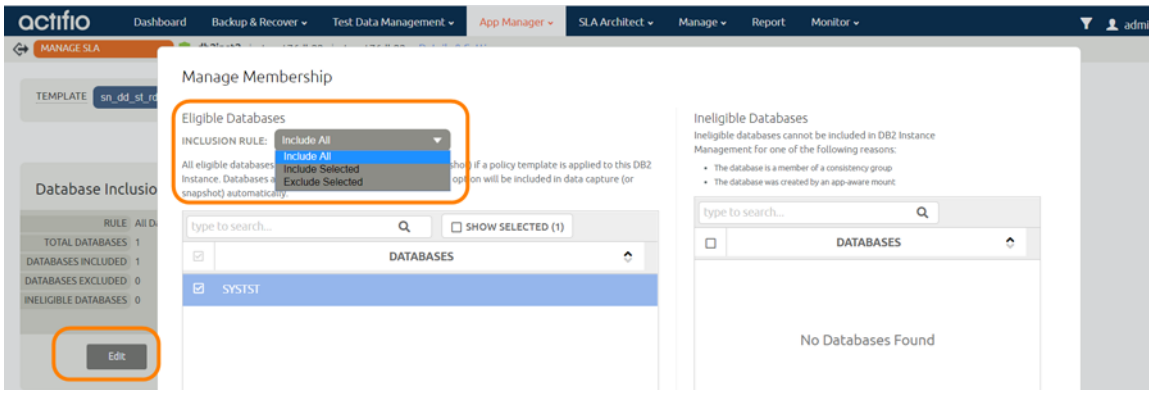


The database will be protected when the snapshot job runs according to the schedule in the template. After the first successful snapshot job the database appears in the App Manager with a green shield icon.

4. You can include or exclude specific databases during backup. From the App Manager, Applications list, select the PostgreSQL Instance. You can use the Instance checkbox to filter the list. Select **Manage SLA**.



5. Under Database Inclusion Rule on the left, click **Edit**. If you do not see the Database Inclusion settings, you have a database, not an instance.

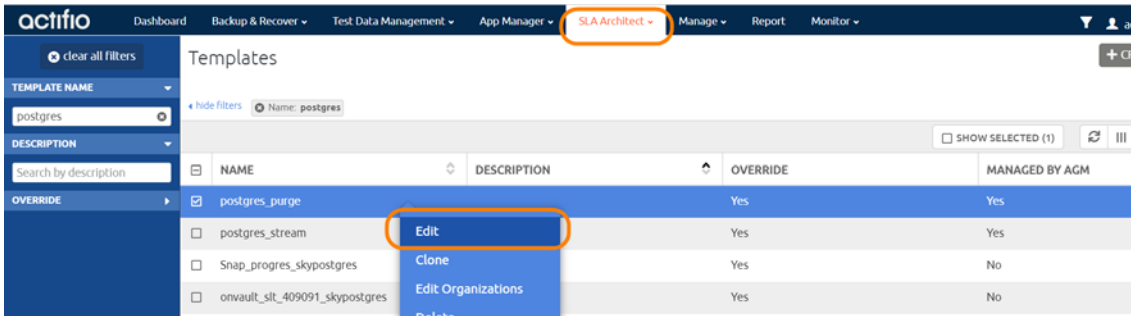


6. Select an Inclusion Rule (Include All, Include Selected, or Exclude Selected) and then select the databases to include or exclude, then click **Save**.

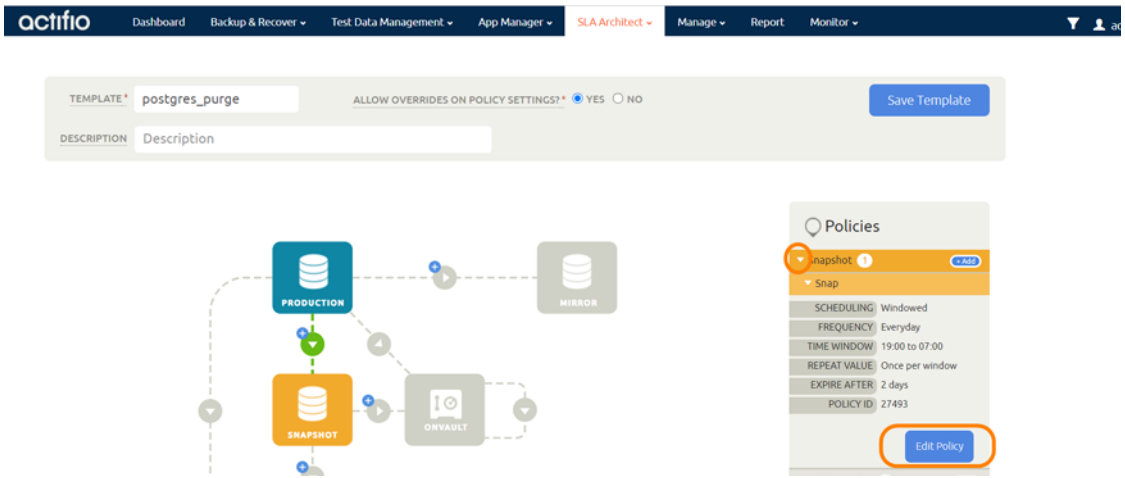
Protecting PostgreSQL Database Logs

To enable PostgreSQL database log backup:

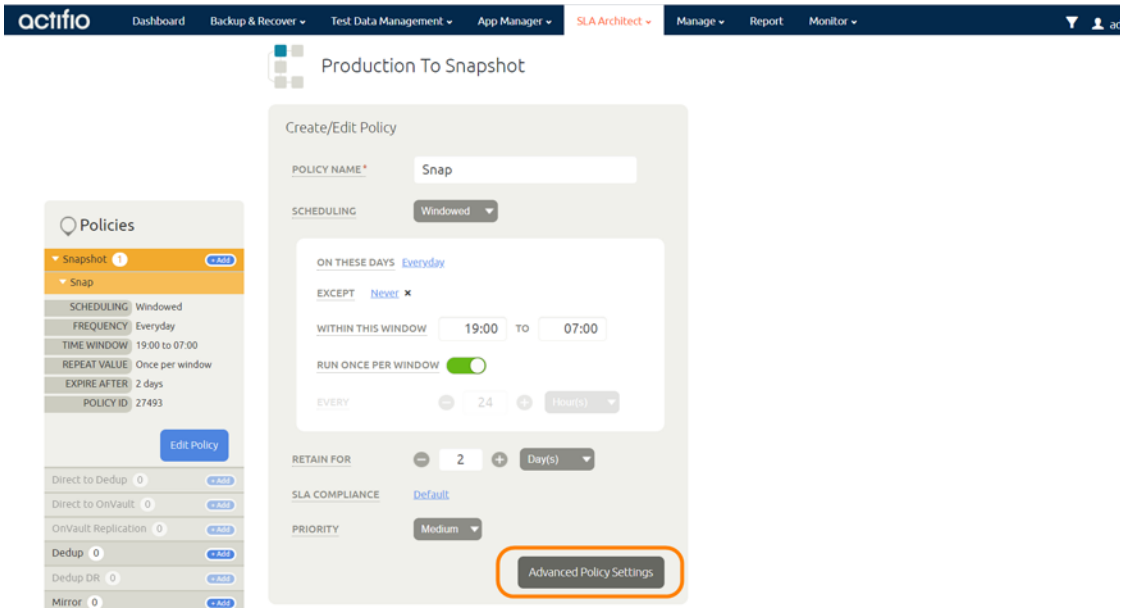
1. From the SLA Architect Templates list, right-click the template for PostgreSQL database protection and click **Edit**.



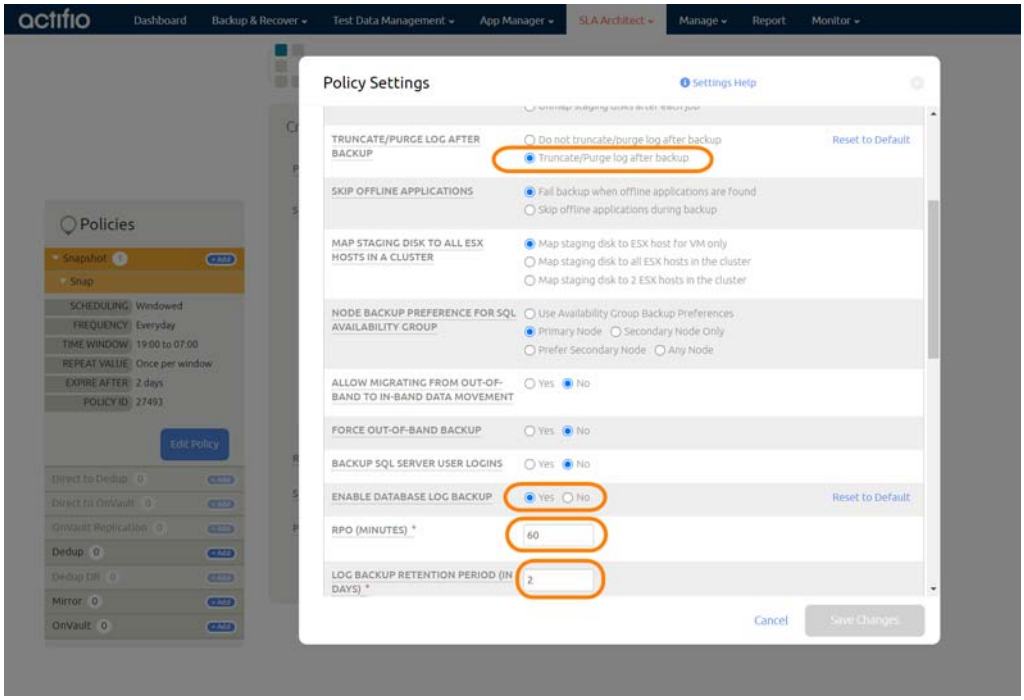
2. Click the arrow beside the Snapshot policy to open up the details, then select **Edit Policy**.



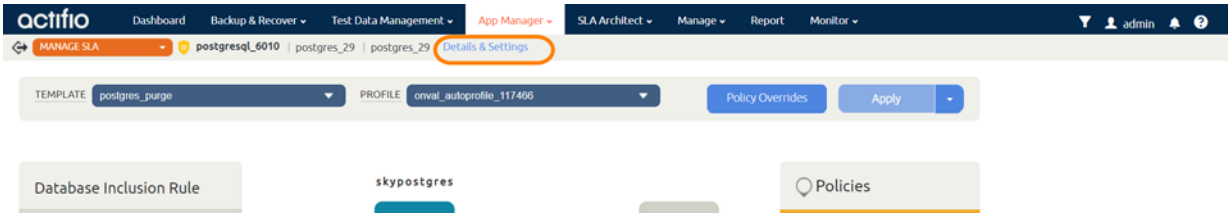
3. Near the bottom of the Create/Edit Policy page, select **Advanced Policy Settings**.



4. Set the log policy options (you will have to scroll to see them all):
   - Enable **Truncate/Purge log after backup**.
   - Set **Enable Database Log Backup** to **Yes**.
   - For **RPO (Minutes)**, enter the desired frequency of log backup.
   - Set **Log Backup Retention Period (in Days)** for point in time recovery.
   - Set **Replicate Logs (Uses StreamSnap Technology)** to **Yes** if you want to enable StreamSnap replication of log backup to a DR site.
   - Set **Log Staging Disk Growth Size** to a percent value that reflects your anticipated usage.



5. Click **Save Changes**.
6. From the App Manager, Applications list, right-click the PostgreSQL Instance and select **Manage SLA**.
7. At the top of the screen, select **Details & Settings**.



8. Set the **Retention of Production DB Logs in Days**. This value is used to purge the logs from the production destination. Based on this setting the log will be purged older then the # of days specified. Default value is 0 days. With the default value, all logs prior to last log backups are purged.
9. Click **Save**.

5 Restoring, Accessing, or Recovering a PostgreSQL Database

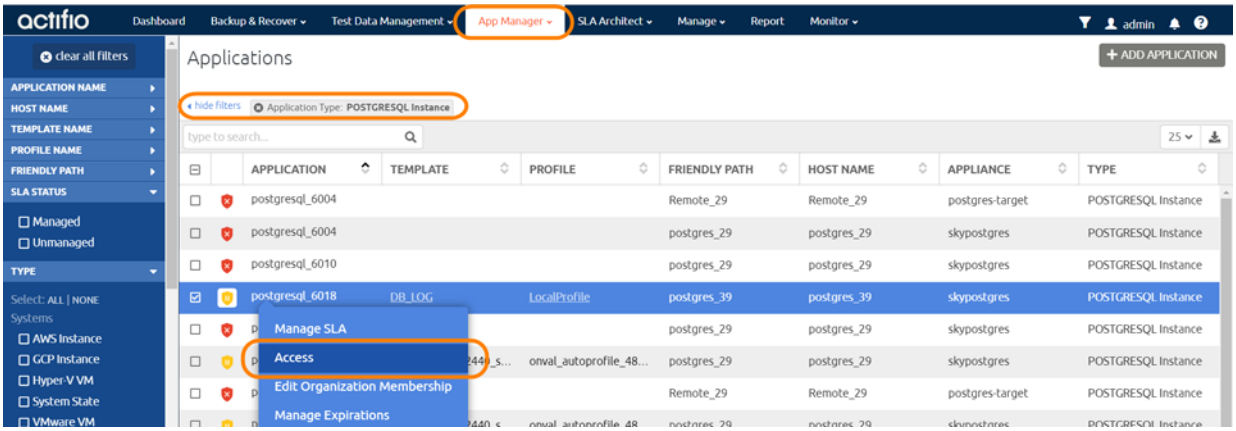
This section describes:

Mounting and Refreshing from Block-Based Volume Snapshot to a Target PostgreSQL Server as a Virtual Application

To mount the database image as a virtual application (an application aware mount) to a new target:

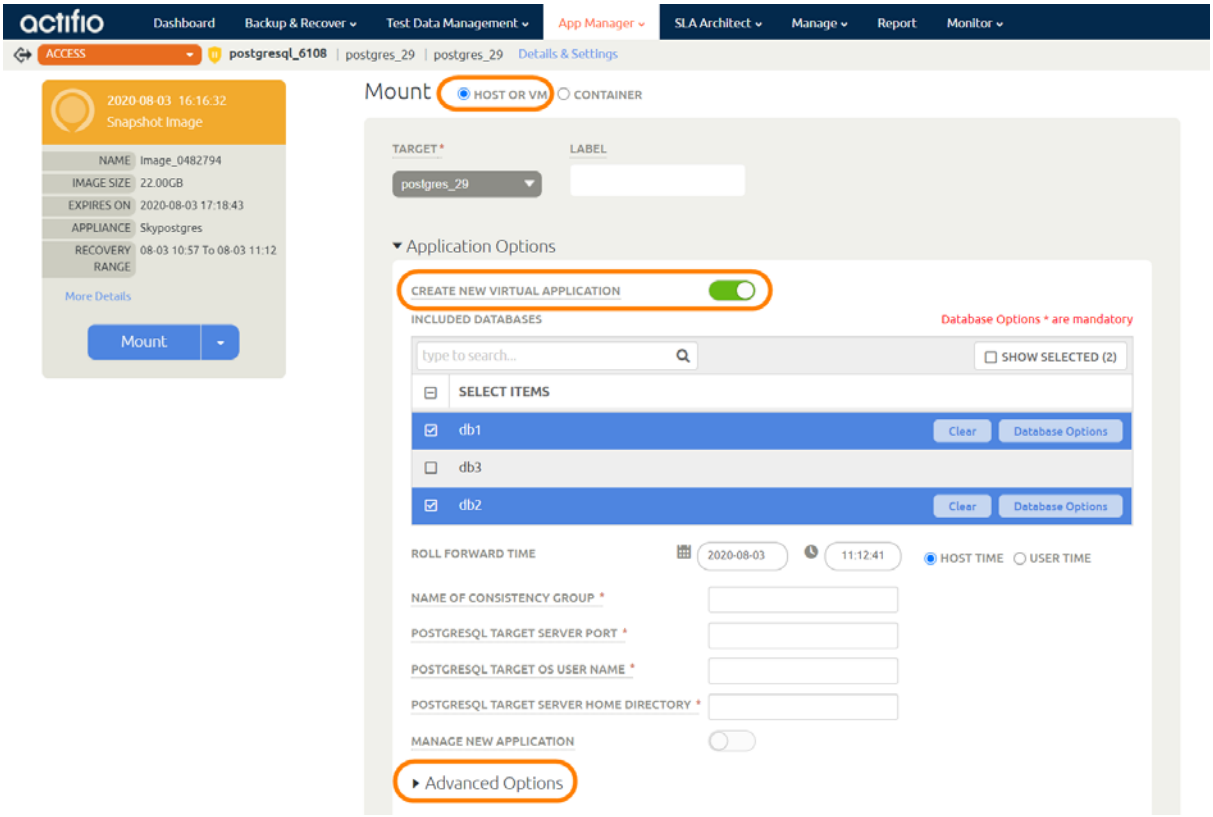
1. From the App Manager, Applications list, right-click the database and select **Access**.

> ***Note:*** *You can use the Managed SLA Status filter to show only protected databases.*



2. Select a snapshot image and choose **Mount**.

3. On the Mount page, from Target, choose the desired target PostgreSQL server port number from the dropdown.
4. Select **Host or VM**, not Container.
5. Under Application Options, enable **Create New Virtual Application**.
6. At Included Databases, Select Items, choose one or more databases to virtualize:
    - A single database will be managed as standalone virtual copy
    - Multiple databases will be managed as a consistency group



7. Click **Database Options** for each selected database to specify a target database name for the new virtual copy.
8. Choose a target point in time for a database protected with log roll-forward.
9. Enter details for the new database/consistency group:

   Name of Consistency Group: This option will appear if more than one database is selected. Provide a unique name to manage the selected databases as a virtual copy.

   PostgreSQL Target Server Port: Enter an unused port number on the target server where the new PostgreSQL instance will be created for the new child databases.

   PostgreSQL Target OS User Name: Enter the name of the operating system user on the target server where the new POSTGRESQL Instance will get created.

   PostgreSQL Target Server Home Directory: Enter the path to the base directory where the configuration files for POSTGRESQL Instance on the target server are stored.

   Manage New Application: To protect the new virtual database, click and enable Manage New Application. Choose a template and a resource profile to protect the database.
10. Use **Advanced Options** to enter credentials and location of the target database messages directory. These are optional.
    - **PostgreSQL Target DB User Name** and **PostgreSQL Target DB Password**: New credentials for the target PostgreSQL Instance that will be created. If you do not specify anything, empty database credentials will be used. By default, a password is not required to log in from the local system.
    - For the **Directory Path**, enter the path to the messages directory for the PostgreSQL Instance on the target server.
    - **Snatch Port by Stopping Existing Instance** specifies whether to stop the existing instance and snatch the port if the target port is already in use by an existing instance.
11. Under Mapping Options:
    - **Storage Pool**: Select a local or external storage pool for the mounted database.
    - **Mount Location**: Specify a target mount point to mount the new virtual database to.
12. Click **Submit**.

Restoring and Recovering a PostgreSQL Instance Back to the Source

Depending on how you protected the database, you need the procedure for:

Recovering from Block-Based Volume Snapshot with CBT on page 20
Recovering from a Full+Incremental Database Backup on page 22

Recovering from Block-Based Volume Snapshot with CBT

Use this procedure to restore and recover the source PostgreSQL instance. This procedure uses physical recovery of the source data area. With this method you can roll-forward protected logs to a point in time.

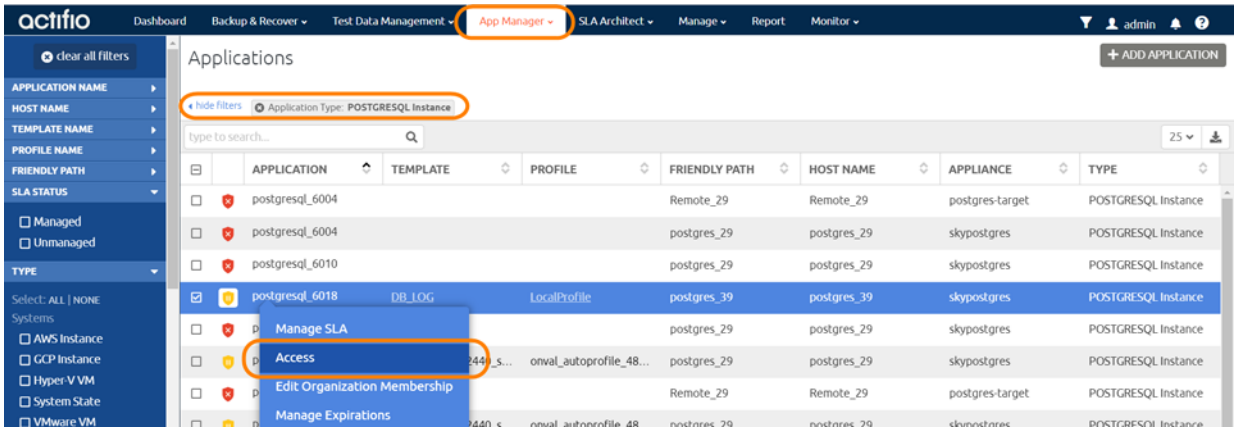
Note: *System databases on a root partition backed up as LVM Snapshots can be mounted as virtual databases, but they cannot be used in a traditional Restore operation as the root partition cannot be unmounted. This will need manual restore and recovery from a simple mount back to the same host.*

Note: *To recover a block-based database image with less downtime for users, see* Mounting and Migrating a PostgreSQL Database for Near-Zero Downtime Recovery to the Source *on page 23.*

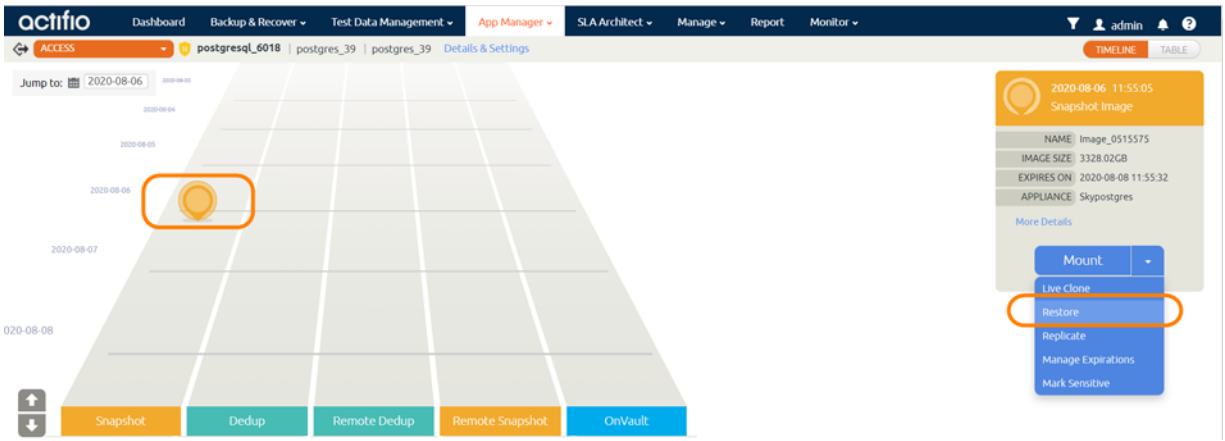
To recover back to the source:

1. From the App Manager, Applications list right-click the protected instance and select **Access**.

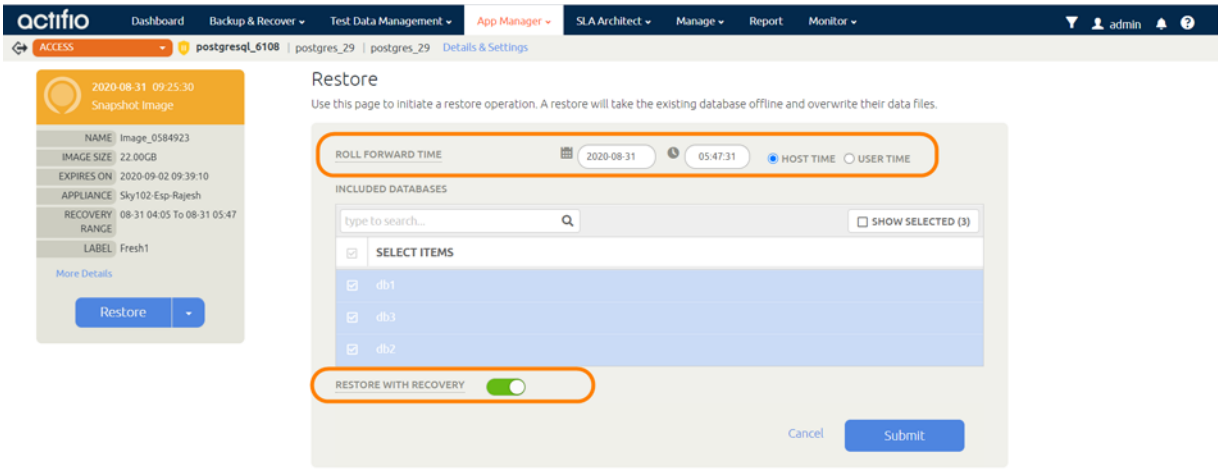
Note: *You can use the Managed SLA Status filter to show only protected instances.*



2. Select a snapshot image and choose **Restore**.



3. Use the **Roll Forward Time** to set the time of the latest good logs.
4. **Select Items** is not available for volume-based images.



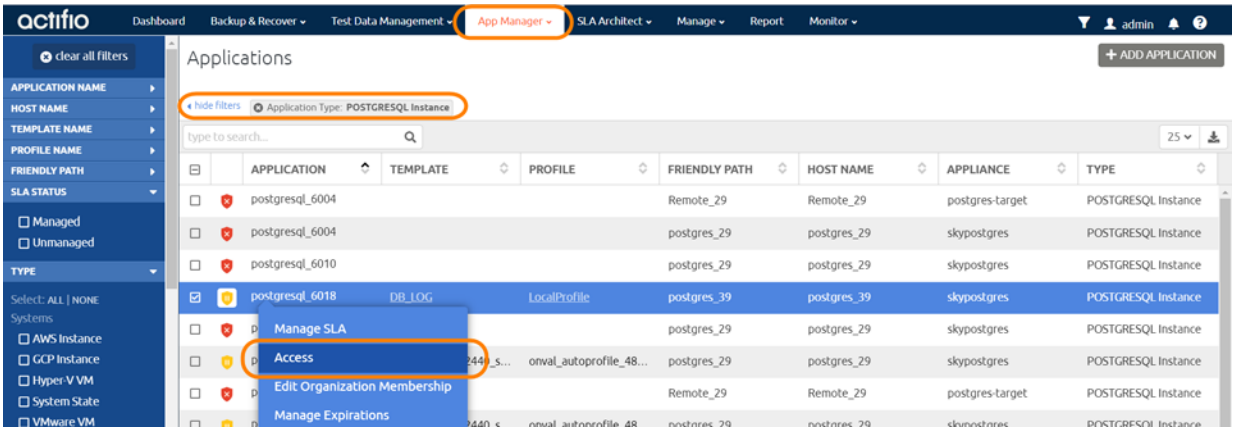
5. Enable **Restore With Recovery** to apply recovered logs to the Roll-Forward time selected above.
6. Click **Submit**.

Recovering from a Full+Incremental Database Backup

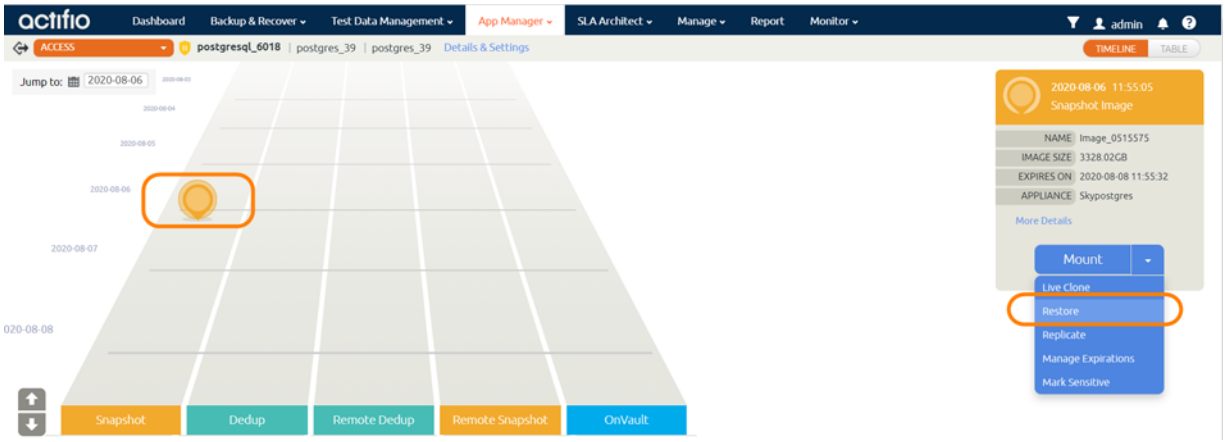
Use this procedure to restore and recover the source database. This overwrites the source data. Log roll-forward to a point-in-time is not supported for Full+Incremental images.

1. From the App Manager, Applications list, right-click the protected database and select **Access**.

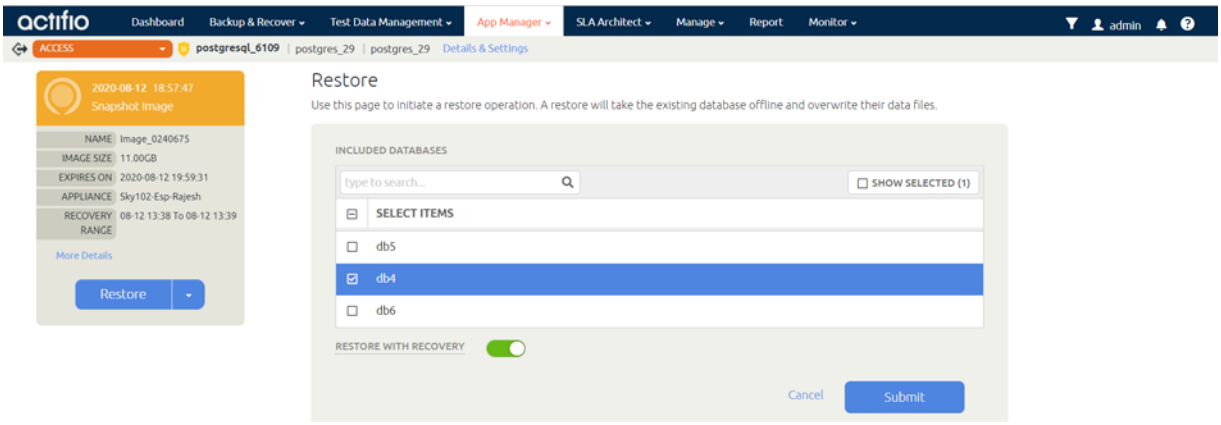
> ***Note:*** *You can use the Managed SLA Status filter to show only protected databases.*



2. Select a snapshot image and choose **Restore**.



3. Use **Select Items** to choose one or more databases to restore.



4. Enable **Restore With Recovery** to apply all recovered logs.
5. Click **Submit**. This will start the source database physical recovery.

Mounting and Migrating a PostgreSQL Database for Near-Zero Downtime Recovery to the Source

A Mount and Migrate operation allows you to restore an application with near-zero downtime by first mounting it locally, and then migrating it to the original location or to a new location. Users have normal access to the application while it is mounted, and the migration step is very fast. This is similar to a VMware Storage VMotion operation.

You can recover SQL Server and file system data instantly using Actifio's existing capabilities, and then migrate the data in real-time to production storage, while the database is up and running.

To mount and migrate a database to a new target, see Mounting and Migrating a PostgreSQL Database to a New Target on page 26.

To recover a database to the source by mounting and migrating:

1. Shut down the source database.
2. Login to AGM, select the application and create a virtual database from a good backup image as detailed in Mounting and Refreshing from Block-Based Volume Snapshot to a Target PostgreSQL Server as a Virtual Application on page 17.
   When setting up the virtual database, provide the source port number for PostgreSQL Target Server Port.
3. When the mount job is completed, run this script, with parameters in Arguments to the Script.

   ```
   /act/custom_apps/postgresql/restore/ACT_POSTGRESQL_lvm_migrate_newTarget.sh
   DATAVOL_DISK_MAPPING=<Actifio_Mount>:<Production_LVM_device> BASEDIR=<BASEDIR>
   OSUSER=<OSUSER> PORT=<PORT> [ DBUSER=<DBUSER> ] [ DBPASSWORD=<DBPASSWORD> ] [
   JOBID=<JOBID> ]
   ```

Arguments to the Script

DATAVOL_DISK_MAPPING=Comma separated list of<Actifio_mount_point>:<equivalent target host lvm device name>

BASEDIR=Target instance PostgreSQL home location

OSUSER=Target database osuser for PostgreSQL

PORT=Target instance port number, which is given during appaware mount

DBUSER=Target instance db username

DBPASSWORD=Target instance password

JOBID=Actifio job id (AppAware mount)

Example

```
/act/custom_apps/postgresql/restore/ACT_POSTGRESQL_lvm_migrate_newTarget.sh
DATAVOL_DISK_MAPPING=/chtst/pgData10.3:/dev/mapper/actdevdatapg103_1594980385483-
act_staging_vol BASEDIR=/home/postgres/postgresql_home_10.3 OSUSER=postgres PORT=6010
DBUSER=postgres JOBID=Job_0957580
```

Note: *The target LVM devices must be empty.*

---

4. Once the script has completed successfully, Unmount and Delete the Image.

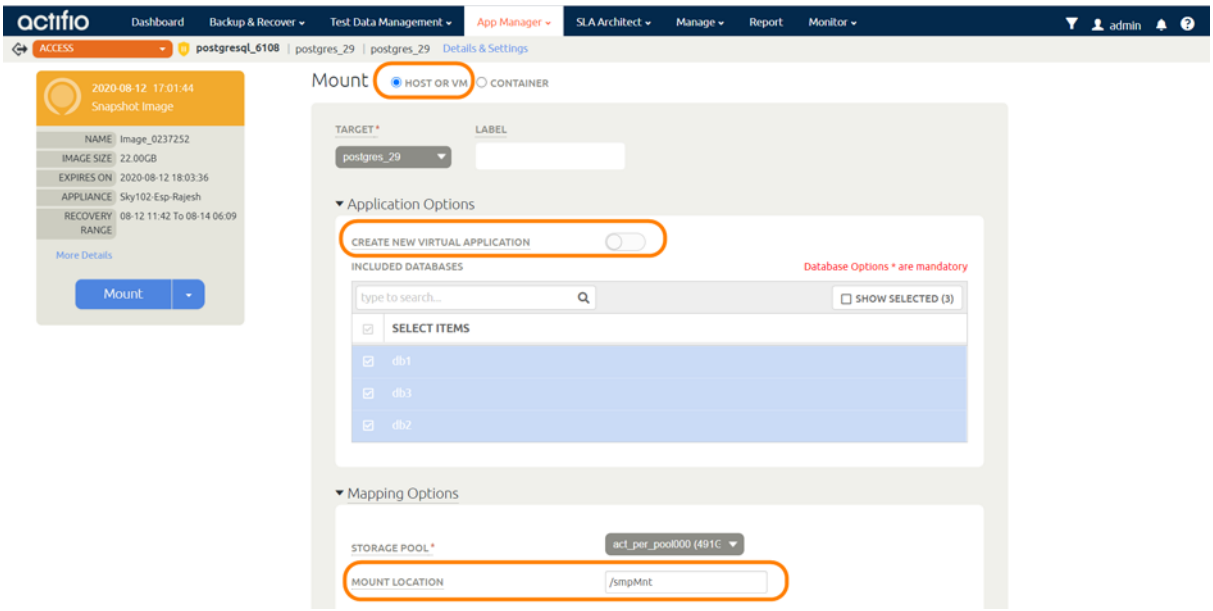
Restoring PostgreSQL Databases to a New Target Using a Block-Based Volume Snapshot

See Configure a WAL Archive on a Secondary Server on page 27

To restore a dump-based backup to a new target:

1. Mount the image as detailed in Mounting and Refreshing from Block-Based Volume Snapshot to a Target PostgreSQL Server as a Virtual Application on page 17. Provide a mount point location but do **not** enable Create Virtual Database.

2. On the Mount page, from Target, choose the desired target server from the dropdown.
3. Select **Host or VM**, not Container.
4. Under Application Options, **disable** Create New Virtual Application.



5. Under Mapping Options:
   - **Storage Pool**: Select a local or external storage pool for the mounted database.
   - **Mount Location**: Specify a target mount point to mount the restored database to.
6. Click **Submit**.

7. Login to the database server as root. On the server, change the directory to /act/custom_apps/postgresql/dump

```
#cd /act/custom_apps/postgresql/dump
```

8. Run this script from the command line (as root) ACT_POSTGRESQL_dumprestore_newTarget.sh on the target with arguments in Arguments to the Script on page 25.

```
/act/custom_apps/postgresql/dump/ACT_POSTGRESQL_dumprestore_newTarget.sh
OSUSER=<postgres_osuser> BASEDIR=<postgres_home> PORT=<postgres_port> DB_LIST=<comma
separated db list> DUMPBKPLOC=<mountpoint name> [ DBUSER=<db_user>
DBPASSWD=<db_password> ]
```

Arguments to the Script

OSUSER = <PostgreSQL instance OS user>

BASEDIR = <PostgreSQL software home location>

PORT = <Target PostgreSQL instance port number>

DB_LIST = <Comma separated database list, which need to restore>

DUMPBKPLOC = <Mountpoint provided during mount>

DBUSER = <PostgreSQL database username>

DBPASSWD = <PostgreSQL database user password>

Example

```
#/act/custom_apps/postgresql/dump/ACT_POSTGRESQL_dumprestore_newTarget.sh
OSUSER=postgres BASEDIR=/home/postgres/postgresql_home_11.0 PORT=5434
DB_LIST=actdb,test1 DUMPBKPLOC=/smpMnt
```

9. Connect to PostgreSQL instance and check if the database was recovered.

```
[postgres@slave.postgres /home/postgres]$ psql -p5434 -Upostgres -dpostgres
psql (11.0)
Type "help" for help.

postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 actdb     | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 hari      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 test1     | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(6 rows)
```

10. Unmount and Delete the Image.

Mounting and Migrating a PostgreSQL Database to a New Target

See Configure a WAL Archive on a Secondary Server on page 27.

A Mount and Migrate operation allows you to restore an application with near-zero downtime by first mounting it locally, and then migrating it to the original location or to a new location. Users have normal access to the application while it is mounted, and the migration step is very fast. This is similar to a VMware Storage VMotion operation. You can move recovered data into other local or SAN storage while the databases are up and running, completing the process with almost no downtime.

To mount and migrate the database to the source, go to Mounting and Migrating a PostgreSQL Database for Near-Zero Downtime Recovery to the Source on page 23.

To create a copy of a database on different storage:

1. Login to AGM, select the database image, and create a virtual database as detailed in Mounting and Refreshing from Block-Based Volume Snapshot to a Target PostgreSQL Server as a Virtual Application on page 17.
2. When the mount job is completed, run this script, with parameters in Arguments to the Script.

```
/act/custom_apps/postgresql/restore/ACT_POSTGRESQL_lvm_migrate_newTarget.sh
DATAVOL_DISK_MAPPING=<Actifio_Mount>:<Production_LVM_device> BASEDIR=<BASEDIR>
OSUSER=<OSUSER> PORT=<PORT> [ DBUSER=<DBUSER> ] [ DBPASSWORD=<DBPASSWORD> ] [
JOBID=<JOBID> ]
```

Arguments to the Script

DATAVOL_DISK_MAPPING=Comma separated list of<Actifio_mount_point>:<equivalent target host lvm device name>

BASEDIR=Target instance PostgreSQL home location

OSUSER=Target database osuser for PostgreSQL

PORT=Target instance port number, which is given during appaware mount

DBUSER=Target instance db username

DBPASSWORD=Target instance password

JOBID=Actifio job id (AppAware mount)

Example

```
/act/custom_apps/postgresql/restore/ACT_POSTGRESQL_lvm_migrate_newTarget.sh
DATAVOL_DISK_MAPPING=/chtst/pgData10.3:/dev/mapper/actdevdatapg103_1594980385483-
act_staging_vol BASEDIR=/home/postgres/postgresql_home_10.3 OSUSER=postgres PORT=6010
DBUSER=postgres JOBID=Job_0957580
```

Note: *The target LVM devices must be empty.*

3. Once the above script completed successfully, Unmount and Delete the Image.

Configure a WAL Archive on a Secondary Server

If you are recovering to a secondary server:

1. Create the directory to keep the WALs. As root user, run:

   ```
   mkdir /<directory>
   chown -R postgres:postgres /<directory>
   ```

2. 1. Configure the parameters for archiving. Run as postgres user:

   Example assumes $PGDATA is /pgdata/11/data.
   Add/update the parameters in the file /pgdata/11/data/postgresql.conf.

   ```
   wal_level = replica
   archive_mode = always
   archive_command = 'test ! -f /pglog/%f && cp %p /pglog/%f'
   ```

3. Restart the PostgreSQL:

   ```
   /usr/pgsql-11/bin/pg_ctl stop -D /pgdata/11/data
   /usr/pgsql-11/bin/pg_ctl start -D /pgdata/11/data
   ```

4. Add/update entry in pg_hba.conf on the primary to accept database connection from the standby.

   Syntax: `host postgres <db-user> <standby-ip/32> <connection method>`

   Example: `host postgres postgres 10.128.0.29/32 trust`

5. Test the archive generation (as postgres user)

   ```
   ls -l /pglog/
   ```

   Then run this log switch command:

   ```
   /usr/pgsql-11/bin/psql -h 10.128.0.28 -p5432 -c "select pg_switch_wal();"
   sleep 5
   ```

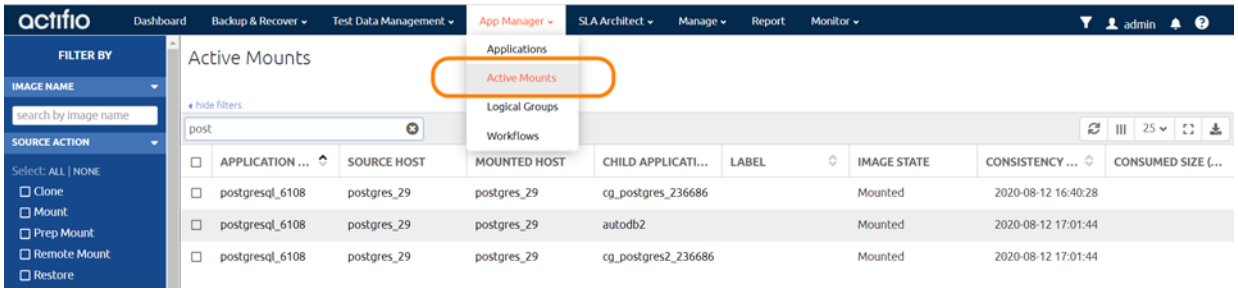
   Log shipping may take some time depending on network

   ```
   ls -l /pglog/
   ```

Unmount and Delete the Image

When you are finished with a mounted image, unmount and delete it to save system resources.

1. From the AGM App Manager, select **Active Mounts**.



2. Right-click the database that the image is from and select **Unmount+Delete**.