# Open Source Databases

## NetApp Solutions

NetApp
October 20, 2023

# Table of Contents

Open Source Databases . . . . . 1
TR-4956: Automated PostgreSQL High Availability Deployment and Disaster Recovery in AWS FSx/EC2 . . 1
TR-4722: MySQL Database on NetApp ONTAP Best Practices . . . . . 10

# Open Source Databases

## TR-4956: Automated PostgreSQL High Availability Deployment and Disaster Recovery in AWS FSx/EC2

Allen Cao, Niyaz Mohamed, NetApp

### Purpose

PostgreSQL is a widely used open-source database that is ranked number four among the top ten most popular database engines by DB-Engines. On one hand, PostgreSQL derives its popularity from its license-free, open-source model while still possessing sophisticated features. On the other hand, because it is open sourced, there is shortage of detailed guidance on production-grade database deployment in the area of high availability and disaster recovery (HA/DR), particularly in the public cloud. In general, it can be difficult to set up a typical PostgreSQL HA/DR system with hot and warm standby, streaming replication, and so on. Testing the HA/DR environment by promoting the standby site and then switching back to the primary can be disruptive to production. There are well documented performance issues on the primary when read workloads are deployed on streaming hot standby.

In this documentation, we demonstrate how you can do away with an application-level PostgreSQL streaming HA/DR solution and build a PostgreSQL HA/DR solution based on AWS FSx ONTAP storage and EC2 compute instances using storage-level replication. The solution creates a simpler and comparable system and delivers equivalent results when compared with traditional PostgreSQL application-level streaming replication for HA/DR.

This solution is built on proven and mature NetApp SnapMirror storage-level replication technology that is available in AWS-native FSX ONTAP cloud storage for PostgreSQL HA/DR. It is simple to implement with an automation toolkit provided by the NetApp Solutions team. It provides similar functionality while eliminating the complexity and performance drag on the primary site with the application-level streaming-based HA/DR solution. The solution can be easily deployed and tested without affecting the active primary site.

This solution addresses the following use cases:

- Production grade HA/DR deployment for PostgreSQL in the public AWS cloud
- Testing and validating a PostgreSQL workload in the public AWS cloud
- Testing and validating a PostgreSQL HA/DR strategy based on NetApp SnapMirror replication technology

### Audience

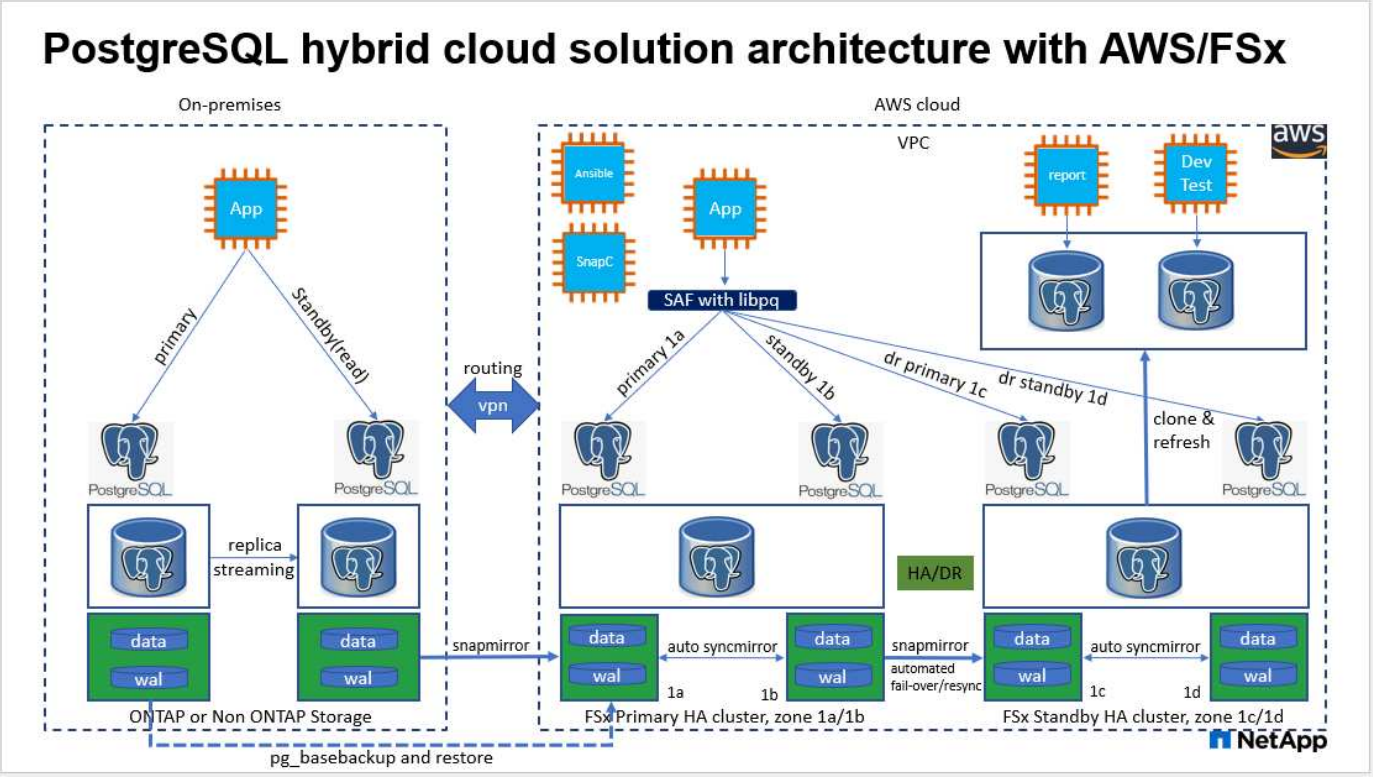
This solution is intended for the following people:

- The DBA who is interested in deploying PostgreSQL with HA/DR in the public AWS cloud.
- The database solution architect who is interested in testing PostgreSQL workloads in the public AWS cloud.
- The storage administrator who is interested in deploying and managing PostgreSQL instances deployed to AWS FSx storage.
- The application owner who is interested in standing up a PostgreSQL environment in AWS FSx/EC2.

## Solution test and validation environment

The testing and validation of this solution was performed in an AWS FSx and EC2 environment that might not match the final deployment environment. For more information, see the section [Key Factors for Deployment Consideration].

### Architecture




### Hardware and software components

| | | |
|---|---|---|
| **Hardware** | | |
| FSx ONTAP storage | Current version | Two FSx HA pairs in the same VPC and availability zone as primary and standby HA clusters |
| EC2 instance for compute | t2.xlarge/4vCPU/16G | Two EC2 T2 xlarge as primary and standby compute instances |
| Ansible controller | on-prem Centos VM/4vCPU/8G | A VM to host Ansible automation controller either on-premise or in the cloud |
| **Software** | | |
| RedHat Linux | RHEL-8.6.0_HVM-20220503-x86_64-2-Hourly2-GP2 | Deployed RedHat subscription for testing |
| Centos Linux | CentOS Linux release 8.2.2004 (Core) | Hosting Ansible controller deployed in on-premises lab |

| | | |
|---|---|---|
| PostgreSQL | Version 14.5 | Automation pulls the latest available version of PostgreSQL from the postgresql.ora yum repo |
| Ansible | Version 2.10.3 | Prerequisites for required collections and libraries installed with requirements playbook |

**Key factors for deployment consideration**

- **PostgreSQL database backup, restore, and recovery.** A PostgreSQL database supports a number of backup methods, such as a logical backup using pg_dump, a physical online backup with pg_basebackup or a lower-level OS backup command, and storage-level-consistent snapshots. This solution uses NetApp consistency-group snapshots for PostgreSQL database data and WAL volumes backup, restore, and recovery at the standby site. The NetApp consistency-group volume snapshots sequence I/O as it is written to storage and protect the integrity of database data files.
- **EC2 compute instances.** In these tests and validations, we used the AWS EC2 t2.xlarge instance type for the PostgreSQL database compute instance. NetApp recommends using an M5 type EC2 instance as the compute instance for PostgreSQL in deployment because it is optimized for database workloads. The standby compute instance should always be deployed in the same zone as the passive (standby) file system deployed for the FSx HA cluster.
- **FSx storage HA clusters single- or multi-zone deployment.** In these tests and validations, we deployed an FSx HA cluster in a single AWS availability zone. For production deployment, NetApp recommends deploying an FSx HA pair in two different availability zones. A disaster-recovery standby HA pair for business continuity can be set up in a different region if a specific distance is required between the primary and standby. An FSx HA cluster is alway provisioned in a HA pair that is sync mirrored in a pair of active-passive file systems to provide storage-level redundancy.
- **PostgreSQL data and log placement.** Typical PostgreSQL deployments share the same root directory or volumes for data and log files. In our tests and validations, we have separated PostgreSQL data and logs into two separate volumes for performance. A soft link is used in the data directory to point to the log directory or volume that hosts PostgreSQL WAL logs and archived WAL logs.
- **PostgreSQL service startup delay timer.** This solution uses NFS mounted volumes to store the PostgreSQL database file and WAL log files. During a database host reboot, PostgreSQL service might try to start while the volume is not mounted. This results in database service startup failure. A 10 to 15 seconds timer delay is needed for the PostgreSQL database to start up correctly.
- **RPO/RTO for business continuity.** FSx data replication from primary to standby for DR is based on ASYNC, which means that the RPO depends on the frequency of Snapshot backups and SnapMirror replication. A higher frequency of Snapshot copy and SnapMirror replication reduces the RPO. Therefore, there is a balance between potential data loss in the event of a disaster and incremental storage cost. We have determined that Snapshot copy and SnapMirror replication can be implemented in as low as 5 minute intervals for RPO, and PostgreSQL can generally be recovered at the DR standby site in under a minute for the RTO.
- **Database backup.** After a PostgreSQL database is implemented or migrated into AWS FSx storage from an on-premisses data center, the data is auto-sync mirrored in the FSx HA pair for protection. Data is further protected with a replicated standby site in case of a disaster. For longer-term backup retention or data protection, NetApp recommends using the built-in PostgreSQL pg_basebackup utility to run a full database backup that can be ported to S3 blob storage.

## Solution Deployment

The deployment of this solution can be completed automatically using the NetApp Ansible-based automation

toolkit by following the detailed instructions outlined below.

1. Read the instructions in the automation toolkit READme.md na_postgresql_aws_deploy_hadr.
2. Watch the following video walk through.

Automated PostgreSQL Deployment and Protection

1. Configure the required parameters files (`hosts`, `host_vars/host_name.yml`, `fsx_vars.yml`) by entering user-specific parameters into the template in the relevant sections. Then use the copy button to copy files to the Ansible controller host.

**Prerequisites for automated deployment**

Deployment requires the following prerequisites.

1. An AWS account has been set up, and the necessary VPC and network segments have been created within your AWS account.
2. From the AWS EC2 console, you must deploy two EC2 Linux instances, one as the primary PostgreSQL DB server at the primary and one at the standby DR site. For compute redundancy at the primary and standby DR sites, deploy two additional EC2 Linux instances as standby PostgreSQL DB servers. See the architecture diagram in the previous section for more details about the environment setup. Also review the User Guide for Linux instances for more information.
3. From the AWS EC2 console, deploy two FSx ONTAP storage HA clusters to host the PostgreSQL database volumes. If you are not familiar with the deployment of FSx storage, see the documentation Creating FSx for ONTAP file systems for step-by-step instructions.
4. Build a Centos Linux VM to host the Ansible controller. The Ansible controller can be located either on-premises or in the AWS cloud. If it is located on-premises, you must have SSH connectivity to the VPC, EC2 Linux instances, and FSx storage clusters.
5. Set up the Ansible controller as described in the section "Set up the Ansible Control Node for CLI deployments on RHEL/CentOS" from the resource Getting Started with NetApp solution automation.
6. Clone a copy of the automation toolkit from the public NetApp GitHub site.

```
git clone https://github.com/NetApp-
Automation/na_postgresql_aws_deploy_hadr.git
```

1. From the toolkit root directory, execute the prerequisite playbooks to install the required collections and libraries for the Ansible controller.

```
ansible-playbook -i hosts requirements.yml
```

```
ansible-galaxy collection install -r collections/requirements.yml --force
--force-with-deps
```

1. Retrieve the required EC2 FSx instance parameters for the DB host variables file `host_vars/*` and the global variables file `fsx_vars.yml` configuration.

**Configure the hosts file**

Input the primary FSx ONTAP cluster management IP and EC2 instances hosts names into the hosts file.

```
# Primary FSx cluster management IP address
[fsx_ontap]
172.30.15.33
```

```
# Primary PostgreSQL DB server at primary site where database is
initialized at deployment time
[postgresql]
psql_01p ansible_ssh_private_key_file=psql_01p.pem
```

```
# Primary PostgreSQL DB server at standby site where postgresql service is
installed but disabled at deployment
# Standby DB server at primary site, to setup this server comment out
other servers in [dr_postgresql]
# Standby DB server at standby site, to setup this server comment out
other servers in [dr_postgresql]
[dr_postgresql] --
psql_01s ansible_ssh_private_key_file=psql_01s.pem
#psql_01ps ansible_ssh_private_key_file=psql_01ps.pem
#psql_01ss ansible_ssh_private_key_file=psql_01ss.pem
```

**Configure the host_name.yml file in the host_vars folder**

```
# Add your AWS EC2 instance IP address for the respective PostgreSQL
server host
ansible_host: "10.61.180.15"

# "{{groups.postgresql[0]}}" represents first PostgreSQL DB server as
defined in PostgreSQL hosts group [postgresql]. For concurrent multiple
PostgreSQL DB servers deployment, [0] will be incremented for each
additional DB server. For example,  "{{groups.posgresql[1]}}" represents
DB server 2, "{{groups.posgresql[2]}}" represents DB server 3 ... As a
good practice and the default, two volumes are allocated to a PostgreSQL
DB server with corresponding /pgdata, /pglogs mount points, which store
PostgreSQL data, and PostgreSQL log files respectively. The number and
naming of DB volumes allocated to a DB server must match with what is
defined in global fsx_vars.yml file by src_db_vols, src_archivelog_vols
parameters, which dictates how many volumes are to be created for each DB
server. aggr_name is aggr1 by default. Do not change. lif address is the
NFS IP address for the SVM where PostgreSQL server is expected to mount
its database volumes. Primary site servers from primary SVM and standby
servers from standby SVM.
host_datastores_nfs:
  - {vol_name: "{{groups.postgresql[0]}}_pgdata", aggr_name: "aggr1", lif:
"172.21.94.200", size: "100"}
  - {vol_name: "{{groups.postgresql[0]}}_pglogs", aggr_name: "aggr1", lif:
"172.21.94.200", size: "100"}

# Add swap space to EC2 instance, that is equal to size of RAM up to 16G
max. Determine the number of blocks by dividing swap size in MB by 128.
swap_blocks: "128"

# Postgresql user configurable parameters
psql_port: "5432"
buffer_cache: "8192MB"
archive_mode: "on"
max_wal_size: "5GB"
client_address: "172.30.15.0/24"
```

**Configure the global fsx_vars.yml file in the vars folder**

```
#########################################################################
######  PostgreSQL HADR global user configuration variables       ######
######  Consolidate all variables from FSx, Linux, and postgresql ######
#########################################################################

########################################
### Ontap env specific config variables ###
```

```
##########################################

#############################################################################
#########################
# Variables for SnapMirror Peering
#############################################################################
#########################

#Passphrase for cluster peering authentication
passphrase: "xxxxxxx"

#Please enter destination or standby FSx cluster name
dst_cluster_name: "FsxId0cf8e0bccb14805e8"

#Please enter destination or standby FSx cluster management IP
dst_cluster_ip: "172.30.15.90"

#Please enter destination or standby FSx cluster inter-cluster IP
dst_inter_ip: "172.30.15.13"

#Please enter destination or standby SVM name to create mirror
relationship
dst_vserver: "dr"

#Please enter destination or standby SVM management IP
dst_vserver_mgmt_lif: "172.30.15.88"

#Please enter destination or standby SVM NFS lif
dst_nfs_lif: "172.30.15.88"

#Please enter source or primary FSx cluster name
src_cluster_name: "FsxId0cf8e0bccb14805e8"

#Please enter source or primary FSx cluster management IP
src_cluster_ip: "172.30.15.20"

#Please enter source or primary FSx cluster inter-cluster IP
src_inter_ip: "172.30.15.5"

#Please enter source or primary SVM name to create mirror relationship
src_vserver: "prod"

#Please enter source or primary SVM management IP
src_vserver_mgmt_lif: "172.30.15.115"

#############################################################################
#########################
```

```
# Variable for PostgreSQL Volumes, lif - source or primary FSx NFS lif
address
#############################################################################
############################

src_db_vols:
  - {vol_name: "{{groups.postgresql[0]}}_pgdata", aggr_name: "aggr1", lif:
"172.21.94.200", size: "100"}

src_archivelog_vols:
  - {vol_name: "{{groups.postgresql[0]}}_pglogs", aggr_name: "aggr1", lif:
"172.21.94.200", size: "100"}

#Names of the Nodes in the ONTAP Cluster
nfs_export_policy: "default"

#############################################################################
############################
### Linux env specific config variables ###
#############################################################################
############################

#NFS Mount points for PostgreSQL DB volumes
mount_points:
  - "/pgdata"
  - "/pglogs"

#RedHat subscription username and password
redhat_sub_username: "xxxxx"
redhat_sub_password: "xxxxx"

#####################################################
### DB env specific install and config variables ###
#####################################################
#The latest version of PostgreSQL RPM is pulled/installed and config file
is deployed from a preconfigured template
#Recovery type and point: default as all logs and promote and leave all
PITR parameters blank
```

**PostgreSQL deployment and HA/DR setup**

The following tasks deploy the PostgreSQL DB server service and initialize the database at the primary site on the primary EC2 DB server host. A standby primary EC2 DB server host is then set up at the standby site. Finally, DB volume replication is set up from the primary-site FSx cluster to the standby-site FSx cluster for disaster recovery.

1. Create DB volumes on the primary FSx cluster, and set up postgresql on the primary EC2 instance host.

```
ansible-playbook -i hosts postgresql_deploy.yml -u ec2-user --private
-key psql_01p.pem -e @vars/fsx_vars.yml
```

2. Set up the standby DR EC2 instance host.

```
ansible-playbook -i hosts postgresql_standby_setup.yml -u ec2-user
--private-key psql_01s.pem -e @vars/fsx_vars.yml
```

3. Set up FSx ONTAP cluster peering and database volume replication.

```
ansible-playbook -i hosts fsx_replication_setup.yml -e
@vars/fsx_vars.yml
```

4. Consolidate the previous steps into a single-step PostgreSQL deployment and HA/DR setup.

```
ansible-playbook -i hosts postgresql_hadr_setup.yml -u ec2-user -e
@vars/fsx_vars.yml
```

5. For setting up a standby PostgreSQL DB host at either the primary or standby sites, comment out all other servers in the hosts file [dr_postgresql] section and then execute the postgresql_standby_setup.yml playbook with the respective target host (such as psql_01ps or standby EC2 compute instance at primary site). Make sure that a host parameters file such as `psql_01ps.yml` is configured under the `host_vars` directory.

```
[dr_postgresql] --
#psql_01s ansible_ssh_private_key_file=psql_01s.pem
psql_01ps ansible_ssh_private_key_file=psql_01ps.pem
#psql_01ss ansible_ssh_private_key_file=psql_01ss.pem
```

```
ansible-playbook -i hosts postgresql_standby_setup.yml -u ec2-user
--private-key psql_01ps.pem -e @vars/fsx_vars.yml
```

**PostgreSQL database snapshot backup and replication to standby site**

PostgreSQL database snapshot backup and replication to the standby site can be controlled and executed on the Ansible controller with a user-defined interval. We have validated that the interval can be as low as 5 minutes. Therefore, in the case of failure at the primary site, there is 5 minutes of potential data loss if failure occurs right before the next scheduled snapshot backup.

```
*/15 * * * * /home/admin/na_postgresql_aws_deploy_hadr/data_log_snap.sh
```

#### Failover to Standby Site for DR

For testing the PostgreSQL HA/DR system as a DR exercise, execute failover and PostgreSQL database recovery on the primary standby EC2 DB instance on standby site by executing following playbook. In an actually DR scenario, execute the same for an actually failover to DR site.

```
ansible-playbook -i hosts postgresql_failover.yml -u ec2-user --private
-key psql_01s.pem -e @vars/fsx_vars.yml
```

#### Resync Replicated DB volumes after Failover Test

Run resync after the failover test to reestablish database-volume SnapMirror replication.

```
ansible-playbook -i hosts postgresql_standby_resync.yml -u ec2-user
--private-key psql_01s.pem -e @vars/fsx_vars.yml
```

#### Failover from primary EC2 DB server to standby EC2 DB server due to EC2 compute instance failure

NetApp recommends running manual failover or using well-established OS cluster-ware that might require a license.

### Where to find additional information

To learn more about the information that is described in this document, review the following documents and/or websites:

- Amazon FSx for NetApp ONTAP

https://aws.amazon.com/fsx/netapp-ontap/

- Amazon EC2

https://aws.amazon.com/pm/ec2/?trk=36c6da98-7b20-48fa-8225-4784bced9843&sc_channel=ps&s_kwcid=AL!4422!3!467723097970!e!!g!!aws%20ec2&ef_id=Cj0KCQiA54KfBhCKARIsAJzSrdqwQrghn6I71jiWzSeaT9Uh1-vY-VfhJixF-xnv5rWwn2S7RqZOTQ0aAh7eEALw_wcB:G:s&s_kwcid=AL!4422!3!467723097970!e!!g!!aws%20ec2

- NetApp Solution Automation

https://docs.netapp.com/us-en/netapp-solutions/automation/automation_introduction.html

# TR-4722: MySQL Database on NetApp ONTAP Best Practices

Anup Bharti, Manohar Kulkarni, Jeffrey Steiner NetApp

MySQL and its variants, including MariaDB and Percona, are widely used for many enterprise applications. These applications range from global social networking sites and massive e-commerce systems to SMB hosting systems containing thousands of database instances. This document describes the configuration requirements and provides guidance on tuning and storage configuration for deploying MySQL on NetApp®

ONTAP® data management software. To determine whether the environment,configurations, and versions specified in this report support your environment, consult the Interoperability Matrix Tool (IMT).

TR-4722: MySQL Database on NetApp ONTAP Best Practices

## Copyright information



## Trademark information