Open Source-Datenbanken

NetApp Solutions

NetApp
January 30, 2025

Inhalt

Open Source-Datenbanken

TR-4956: Automatisierte PostgreSQL High Availability Implementierung und Disaster Recovery in AWS FSX/EC2

Allen Cao, Niyaz Mohamed, NetApp

Diese Lösung bietet Übersicht und Details zur Implementierung von PostgreSQL-Datenbanken und zur Einrichtung von HA/DR, Failover und Synchronisierung basierend auf NetApp SnapMirror Technologie in FSX ONTAP Storage-Angebot und NetApp Ansible Automatisierungs-Toolkit in AWS.

Zweck

PostgreSQL ist eine häufig verwendete Open-Source-Datenbank, die auf Platz vier unter den zehn beliebtesten Datenbank-Engines von rangiert "DB-Engines". Auf der einen Seite, PostgreSQL leitet seine Popularität aus seinem lizenzfreien, Open-Source-Modell, während noch mit ausgereiften Features. Da es sich um Open-Source-Lösungen handelt, fehlen insbesondere in der Public Cloud detaillierte Leitfäden zur Implementierung produktionsferglicher Datenbanken. Dies gilt für Hochverfügbarkeit und Disaster Recovery (HA/DR). Im Allgemeinen kann es schwierig sein, ein typisches PostgreSQL HA/DR-System mit warmem Standby, Streaming-Replizierung usw. einzurichten. Das Testen der HA/DR-Umgebung durch Beförderung des Standby-Standorts und dann das Zurückschalten zum primären Standort kann mit der Produktion zu Störungen führen. Es gibt auf dem primären Volume gut dokumentierte Performance-Probleme, wenn Lese-Workloads auf dem Streaming des Hot-Standby-Modus ausgeführt werden.

In dieser Dokumentation zeigen wir, wie Sie eine PostgreSQL Streaming HA/DR-Lösung auf Applikationsebene hinter sich lassen und eine PostgreSQL HA/DR-Lösung auf Basis von AWS FSX ONTAP Storage und EC2 Computing-Instanzen mittels Storage-Level-Replizierung aufbauen können. Die Lösung erstellt ein einfacheres, vergleichbares System und liefert ähnliche Ergebnisse im Vergleich zur herkömmlichen Streaming-Replizierung auf PostgreSQL-Applikationsebene für HA/DR.

Diese Lösung basiert auf der bewährten und ausgereiften NetApp SnapMirror Replizierungstechnologie auf Storage-Ebene, die auch in FSX ONTAP Cloud-Storage für PostgreSQL HA/DR in AWS verfügbar ist. Die Implementierung ist mit einem vom NetApp Solutions Team bereitgestellten Automatisierungs-Toolkit einfach. Es bietet ähnliche Funktionen und beseitigt gleichzeitig die Komplexität- und Performance-Einbußen am primären Standort mit der auf Applikationsebene basierenden Streaming-basierten HA/DR-Lösung auf Applikationsebene. Die Lösung kann einfach implementiert und getestet werden, ohne dass der aktive primäre Standort davon betroffen ist.

Diese Lösung eignet sich für folgende Anwendungsfälle:

- HA/DR-Implementierung auf Produktionsniveau für PostgreSQL in der Public AWS Cloud
- Testen und Validieren eines PostgreSQL-Workloads in der Public AWS Cloud
- Testen und Validieren einer PostgreSQL HA/DR-Strategie auf der Basis der NetApp SnapMirror Replizierungstechnologie

Zielgruppe

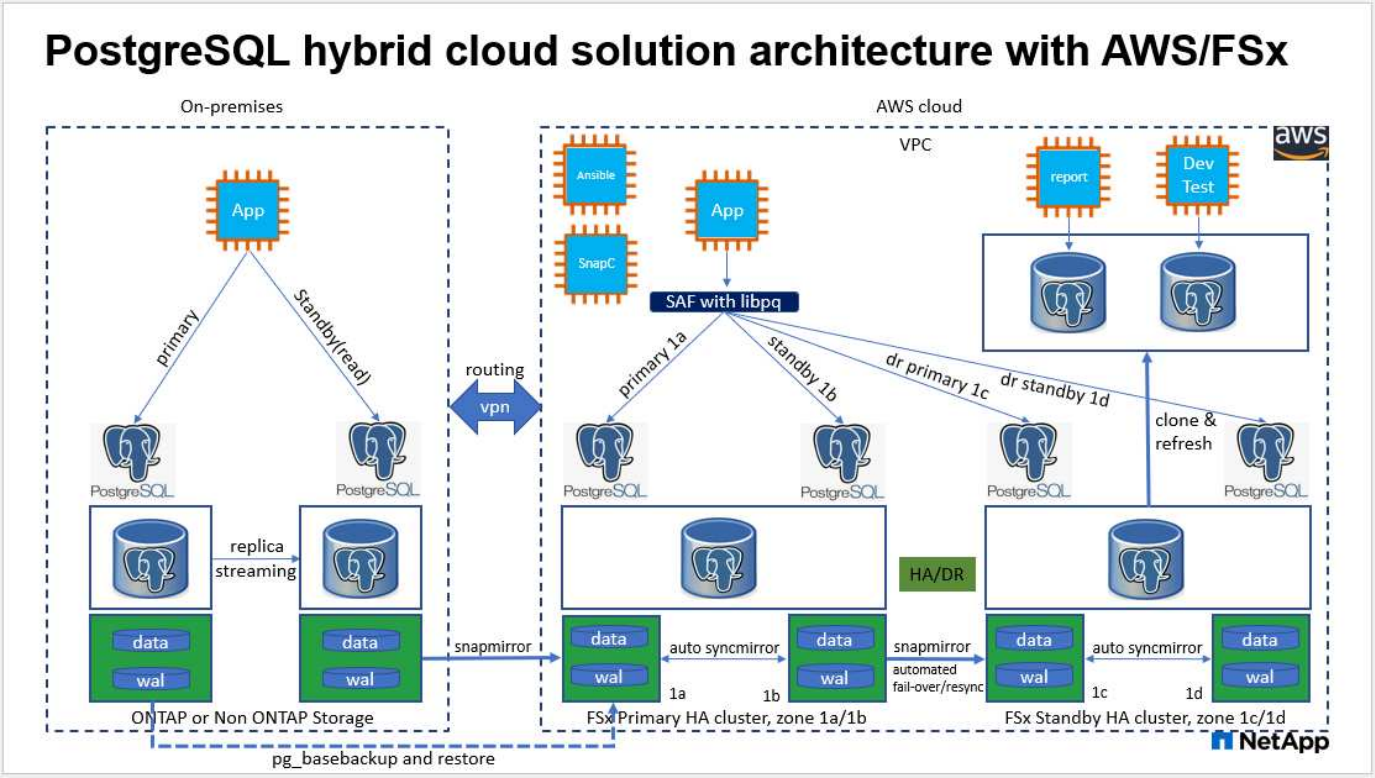
Diese Lösung ist für folgende Personen gedacht:

- Der DBA, der an der Implementierung von PostgreSQL mit HA/DR in der Public AWS Cloud interessiert ist.
- Der Datenbanklösungsarchitekt, der PostgreSQL-Workloads in der Public AWS Cloud testen möchte.
- Storage-Administrator, der an der Implementierung und dem Management von PostgreSQL-Instanzen interessiert ist, die auf AWS FSX Storage bereitgestellt werden.
- Der Applikationseigentümer, der an der Einrichtung einer PostgreSQL-Umgebung in AWS FSX/EC2 interessiert ist.

Test- und Validierungsumgebung der Lösung

Tests und Validierungen dieser Lösung wurden in einer AWS FSX- und EC2-Umgebung durchgeführt, die möglicherweise nicht mit der endgültigen Implementierungsumgebung übereinstimmt. Weitere Informationen finden Sie im Abschnitt Wichtige Faktoren für die Implementierung.

Der Netapp Architektur Sind



Hardware- und Softwarekomponenten

Hardware		
FSX ONTAP-Storage	Aktuelle Version	Zwei FSX HA-Paare in derselben VPC und Verfügbarkeitszone wie primäre und Standby HA-Cluster
EC2 Instanz für Computing	t2.xlarge/4vCPU/16G	Zwei EC2 T2 xlarge als primäre und Standby-Computing-Instanzen
Ansible-Controller	CentOS VM On-Prem./4vCPU/8 G	VM zum Hosten des Ansible-Automatisierungs-Controllers entweder vor Ort oder in der Cloud

Software		
Redhat Linux	RHEL-8.6.0_HVM-20220503-x86_64-2-Hourly2-GP2	Bereitstellung der RedHat Subscription für Tests
Centos Linux	CentOS Linux Version 8.2.2004 (Core)	Hosting des Ansible-Controllers im On-Premises-Lab
PostgreSQL	Version 14.5	Automation zieht die neueste verfügbare Version von PostgreSQL aus der postgresql.ora yum repo
Ansible	Version 2.10.3	Voraussetzungen für erforderliche Sammlungen und Bibliotheken, die mit dem Requirements Playbook installiert sind

Wichtige Faktoren für die Implementierung

- **Sicherung, Wiederherstellung und Wiederherstellung von PostgreSQL-Datenbanken.** Eine PostgreSQL-Datenbank unterstützt eine Reihe von Backup-Methoden wie ein logisches Backup mit pg_dump, ein physisches Online-Backup mit pg_basebackup oder einen niedrigeren Level-Befehl zum Sichern von Betriebssystemen sowie konsistente Snapshots auf Storage-Ebene. Diese Lösung verwendet NetApp Consistency-Group Snapshots für PostgreSQL-Datenbankdaten und WAL Volumes für Backup, Restore und Recovery am Standby-Standort. Die NetApp Consistency Group Volume Snapshots sequenzieren die I/O-Vorgänge, während sie in den Storage geschrieben werden, und schützen die Integrität von Datenbankdatendateien.
- **EC2 Compute-Instanzen.** bei diesen Tests und Validierungen haben wir den AWS EC2 instanztyp für die PostgreSQL-Datenbank-Computing-Instanz verwendet. NetApp empfiehlt die Verwendung einer M5-Typ-EC2-Instanz als Computing-Instanz für PostgreSQL bei der Implementierung, da sie für Datenbank-Workloads optimiert ist. Die Standby-Computing-Instanz sollte immer in derselben Zone wie das passive (Standby) Filesystem, das für das FSX HA-Cluster bereitgestellt wird, bereitgestellt werden.
- **FSX Storage HA Cluster Single- oder Multi-Zone-Implementierung.** bei diesen Tests und Validierungen haben wir einen FSX HA-Cluster in einer einzelnen AWS Verfügbarkeitszone implementiert. Für die Implementierung in der Produktion empfiehlt NetApp die Implementierung eines FSX HA-Paars in zwei verschiedenen Verfügbarkeitszonen. Ein Disaster Recovery-Standby-HA-Paar für Business Continuity kann in einer anderen Region eingerichtet werden, wenn zwischen dem primären und dem Standby eine bestimmte Entfernung erforderlich ist. Ein FSX HA-Cluster wird in einem HA-Paar bereitgestellt, das in einem Paar aktiv/Passiv-Filesysteme gespiegelt wird, um Redundanz auf Storage-Ebene bereitzustellen.
- **PostgreSQL Daten- und Log-Platzierung.** Typische PostgreSQL-Bereitstellungen teilen sich das gleiche Stammverzeichnis oder dieselben Volumes für Daten- und Log-Dateien. Bei unseren Tests und Validierungen haben wir PostgreSQL Daten und Logs zu zwei separaten Volumes für die Leistung getrennt. Ein Soft-Link wird im Datenverzeichnis verwendet, um auf das Logverzeichnis oder Volume zu verweisen, das PostgreSQL WAL-Logs und archivierte WAL-Logs hostet.
- **PostgreSQL Service Startup Delay Timer.** Diese Lösung verwendet NFS gemountete Volumen, um die PostgreSQL Datenbank-Datei und WAL Log-Dateien zu speichern. Während eines Neustart eines Datenbank-Hosts versucht der PostgreSQL-Dienst möglicherweise, zu starten, während das Volume nicht angehängt ist. Dies führt zu einem Fehler beim Starten des Datenbankdienstes. Für den korrekten Start der PostgreSQL-Datenbank ist eine Zeitverzögerung von 10 bis 15 Sekunden erforderlich.
- **RPO/RTO für Business Continuity.** FSX Datenreplikation vom primären zum Standby für DR basiert auf ASYNC, das bedeutet, dass der RPO von der Häufigkeit von Snapshot Backups und SnapMirror Replikation abhängt. Je häufiger Snapshot Kopien und SnapMirror Replizierung erstellt werden, desto

geringer die RPO. Daher gibt es ein Gleichgewicht zwischen potentiellem Datenverlust im Falle eines Notfalls und inkrementellen Storage-Kosten. Wir haben festgestellt, dass Snapshot Kopie und SnapMirror Replizierung in nur 5-Minuten-Intervallen für RPO implementiert werden können und dass PostgreSQL in der Regel innerhalb einer Minute am DR-Standby-Standort wiederhergestellt werden kann.

- **Datenbank-Backup.** Nachdem eine PostgreSQL-Datenbank von einem On-premisses Data Center aus implementiert oder in den AWS FSX-Speicher migriert wurde, werden die Daten zur Absicherung im FSX HA-Paar automatisch gespiegelt. Daten werden im Notfall über einen replizierten Standby-Standort weiter gesichert. Für eine längerfristige Backup-Aufbewahrung oder Datensicherung empfiehlt NetApp die Nutzung des integrierten PostgreSQL pg_baseBackup Utility, um ein vollständiges Datenbank-Backup auszuführen, das auf S3 Blob-Storage portiert werden kann.

Lösungsimplementierung

Die Implementierung dieser Lösung kann mit dem auf NetApp Ansible basierenden Automatisierungs-Toolkit automatisch abgeschlossen werden. Befolgen Sie die detaillierten Anweisungen unten.

1. Lesen Sie die Anweisungen im Automations-Toolkit Readme.md "na_postgresql_aws_Deploy_hadr".
2. Sehen Sie sich das folgende Video an.

Automatisierte PostgreSQL-Implementierung und -Sicherung

1. Konfigurieren Sie die erforderlichen Parameterdateien (`hosts`, `host_vars/host_name.yml`, `fsx_vars.yml`) Durch Eingabe benutzerspezifischer Parameter in die Vorlage in den entsprechenden Abschnitten. Dann kopieren Sie mit der Schaltfläche Kopieren Dateien auf den Ansible-Controller-Host.

Voraussetzungen für die automatisierte Bereitstellung

Die Bereitstellung erfordert die folgenden Voraussetzungen.

1. Es wurde ein AWS Konto eingerichtet, und die erforderlichen VPC und Netzwerksegmente wurden in Ihrem AWS Konto erstellt.
2. Über die AWS EC2-Konsole müssen Sie zwei EC2 Linux-Instanzen bereitstellen, eine als primärer PostgreSQL DB-Server auf dem primären und eine am Standby-DR-Standort. Um Rechenredundanz auf dem primären und Standby-DR-Standort zu erreichen, sollten zwei zusätzliche EC2 Linux Instanzen als Standby PostgreSQL DB Server implementiert werden. Im Architekturdiagramm im vorherigen Abschnitt finden Sie weitere Details zum Umgebungs-Setup. Sehen Sie sich auch die an "Benutzerhandbuch für Linux-Instanzen" Finden Sie weitere Informationen.
3. Implementieren Sie über die AWS EC2 Konsole zwei FSX ONTAP Storage HA-Cluster, um die PostgreSQL Datenbank-Volumes zu hosten. Wenn Sie mit der Implementierung von FSX Storage nicht vertraut sind, finden Sie in der Dokumentation "Erstellen von FSX ONTAP-Dateisystemen"eine Schritt-für-Schritt-Anleitung.
4. Eine CentOS Linux VM aufbauen, um den Ansible-Controller zu hosten. Der Ansible-Controller kann sich entweder vor Ort oder in der AWS Cloud befinden. Falls die Daten lokal gespeichert sind, müssen SSH-Konnektivität mit der VPC, EC2 Linux Instanzen und FSX Storage-Cluster vorhanden sein.
5. Richten Sie den Ansible-Controller wie in dem Abschnitt „Ansible-Steuerungsknoten für CLI-Bereitstellungen auf RHEL/CentOS einrichten“ von der Ressource aus ein "Erste Schritte mit der Automatisierung von NetApp Lösungen".
6. Klonen einer Kopie des Automatisierungs-Toolkit auf der öffentlichen NetApp GitHub Website.

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

1. Führen Sie im Root-Verzeichnis des Toolkit die erforderlichen Playbooks aus, um die für den Ansible Controller erforderlichen Sammlungen und Bibliotheken zu installieren.

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

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

1. Rufen Sie die erforderlichen EC2 FSX-Instanzparameter für die DB-Hostvariablen-Datei ab `host_vars/*` Und die globale Variablendatei `fsx_vars.yml` Konfiguration.

Konfigurieren Sie die Host-Datei

Geben Sie die primäre FSX ONTAP-Cluster-Management-IP und EC2-Instanzen Hostnamen in die Hosts-Datei ein.

```
# 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
```

Konfigurieren Sie die Datei Host_Name.yml im Ordner Host_vars

```
# 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"
```

Konfigurieren Sie die globale fsx_Vars.yml-Datei im Ordner Vars

```
#########################################################################
######  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 Implementierung und HA/DR-Einrichtung

Die folgenden Aufgaben implementieren den PostgreSQL DB Serverdienst und initialisieren die Datenbank am primären Standort auf dem primären EC2 DB-Serverhost. Ein Standby-primären EC2 DB-Server-Host wird dann am Standby-Standort eingerichtet. Schließlich wird die DB-Volume-Replizierung aus dem FSX-Cluster des primären Standorts auf dem FSX-Cluster des Standby-Standorts eingerichtet, um Disaster Recovery zu

ermöglichen.

1. Erstellen Sie DB-Volumes auf dem primären FSX-Cluster und richten sie postgresql auf dem primären EC2-Instance-Host ein.

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

2. Richten Sie den Standby-DR EC2-Instance-Host ein.

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

3. FSX ONTAP-Cluster-Peering und Datenbank-Volume-Replizierung einrichten

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

4. Konsolidieren Sie die vorherigen Schritte in einer PostgreSQL Implementierung mit einem Schritt und HA/DR-Einrichtung.

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

5. Um einen Standby PostgreSQL DB-Host entweder auf dem primären oder Standby-Standort einzurichten, kommentieren Sie alle anderen Server im Abschnitt Hosts-Datei [dr_postgresql] und führen Sie dann das Playbook postgresql_Standby_Setup.yml mit dem jeweiligen Zielhost aus (z. B. psql_01ps oder Standby EC2 Compute-Instanz am primären Standort). Stellen Sie sicher, dass eine Host-Parameterdatei wie z. B. `psql_01ps.yml` Wird unter konfiguriert `host_vars` Verzeichnis.

```
[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
```

Snapshot-Backup und Replikation der PostgreSQL-Datenbank auf Standby-Standort

Die Sicherung und Replikation von PostgreSQL-Datenbank-Snapshots auf den Standby-Standort können auf dem Ansible-Controller mit einem benutzerdefinierten Intervall gesteuert und ausgeführt werden. Wir haben

validiert, dass das Intervall nur 5 Minuten betragen kann. Daher kann bei einem Ausfall am primären Standort direkt vor dem nächsten geplanten Snapshot Backup ein Datenverlust von 5 Minuten auftreten.

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

Failover zum Standby-Standort für DR

Führen Sie zum Testen des PostgreSQL HA/DR-Systems als DR-Übung Failover und Wiederherstellung der PostgreSQL Datenbank auf der primären Standby EC2 DB Instanz am Standby-Standort durch, indem Sie das folgende Playbook ausführen. Führen Sie in einem DR-Szenario die gleiche Ausführung für ein tatsächlicher Failover zum DR-Standort aus.

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

Synchronisieren Sie replizierte DB-Volumes nach Failover-Test

Führen Sie die Resynchronisierung nach dem Failover-Test durch, um die SnapMirror Replikation des Datenbankvolumens wiederherzustellen.

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

Failover vom primären EC2 DB-Server zum Standby-EC2-DB-Server aufgrund des Ausfalls der EC2-Computing-Instanz

NetApp empfiehlt, manuelle Failover-Vorgänge auszuführen oder bewährte Betriebssystem-Cluster-Software zu verwenden, die möglicherweise eine Lizenz erfordern.

Wo Sie weitere Informationen finden

Sehen Sie sich die folgenden Dokumente und/oder Websites an, um mehr über die in diesem Dokument beschriebenen Informationen zu erfahren:

- Amazon FSX 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

- Automatisierung der NetApp Lösung

"Einführung"

TR-4722: Best Practices für MySQL Datenbanken auf NetApp ONTAP

Anup Bharti, Manohar Kulkarni, Jeffrey Steiner NetApp

MySQL und seine Varianten, darunter MariaDB und Percona, sind weit verbreitet für viele Unternehmensanwendungen. Diese Anwendungen reichen von globalen Websites sozialer Netzwerke und massiven E-Commerce-Systemen bis hin zu SMB-Hosting-Systemen mit Tausenden von Datenbankinstanzen. Dieses Dokument beschreibt die Konfigurationsanforderungen und bietet Anleitung zum Tuning und zur Speicherkonfiguration für die Bereitstellung von MySQL auf NetApp® ONTAP® Datenmanagement-Software. Um zu ermitteln, ob die in diesem Bericht angegebene Umgebung, die Konfigurationen und Versionen Ihre Umgebung unterstützen, finden Sie im Interoperabilitäts-Matrix-Tool (IMT).

"TR-4722: Best Practices für MySQL Datenbanken auf NetApp ONTAP"