The science behind the report:

Boost app development and management with VMware Data Services Manager

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Boost app development and management with VMware Data Services Manager.

We concluded our hands-on testing on March 16, 2026. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on March 11, 2026 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to https://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: Results of our testing.

	Time to execute in DSM (mm:ss)	Time to execute manually (mm:ss)	% less time, DSM vs manual	Time savings, DSM vs manual (mm:ss)
Provisioning a new PostgreSQL database/VM				
Run 1	5:10	8:03	-	-
Run 2	5:01	7:24	-	-
Run 3	5:13	7:47	-	-
Median	5:10	7:47	33.6%	2:37
Cloning an existing PostgreSQL database/VM				
Run 1	7:20	10:41	-	-
Run 2	7:41	10:23	-	-
Run 3	7:29	11:02	-	-
Median	7:29	10:41	29.9%	3:12
Updating a PostgreSQL database major version (PG16->17)				
Run 1	2:52	5:04	-	-
Run 2	2:39	5:15	-	-
Run 3	2:44	4:56	-	-
Median	2:44	5:04	46.0%	2:20

	Time to execute in DSM (mm:ss)	Time to execute manually (mm:ss)	% less time, DSM vs manual	Time savings, DSM vs manual (mm:ss)
Provisioning a three-node PostgreSQL HA cluster				
Run 1	09:10	46:42	-	-
Run 2	09:52	44:35	-	-
Run 3	09:08	44:52	-	-
Median	09:10	44:52	79.5%	35:42
Scaling out a one-node PostgreSQL instance to three-node HA cluster				
Run 1	05:52	39:10	-	-
Run 2	05:41	37:08	-	-
Run 3	05:36	37:43	-	-
Median	05:41	37:43	84.9%	32:02

How we tested

Completing the scenarios with DSM

Provisioning a new PostgreSQL 16 database in DSM

1. Log into DSM, and start the timer.
2. Click Databases → Postgres.
3. Click Create Database.
4. Set Version to 16.11+vmware.v9.1.0.0.
5. Input the desired instance name and admin password.
6. Click Next.
7. Select Single Server.
8. Enable Remote Replication.
9. Enter the desired Replication Slot Name.
10. Click Next.
11. Set the VM Class to medium.
12. Set the Disk Size to 40GB.
13. Click Next.
14. Click Next.
15. Click Create. Stop the timer when the database status changes from In Progress to Ready in the Postgres databases listing, and record the result.

Cloning an existing PostgreSQL database in DSM

1. Log into DSM, and start the timer.
2. Click Databases → Postgres.
3. Click the three dots to the left of the instance name for the previously created database, and select Clone.
4. Input the instance name for the cloned database.
5. Click Clone Now.
6. Stop the timer when the cloned database status changes to Ready, and record the result.

Updating PostgreSQL database version from 16 to 17

1. Log into DSM, and start the timer.
2. Click Databases → Postgres.
3. Click the instance name of the previously created database.
4. Scroll to Available Upgrade(s) and History and click Upgrade Now next to target version 17.7+vmware.v9.1.0.0.
5. Click Confirm. Stop the timer when the database status changes from Upgrading to Ready, and record the result.

Completing the scenarios manually

Creating a gold PostgreSQL 16 VM template in VMware vCenter for testing

1. Log into vCenter.
2. Right-click the cluster, and click New Virtual Machine.
3. Click Next.
4. Enter a name for the VM (e.g., gold-ubuntu-pg16), and click Next.
5. Select the destination compute resource, and click Next.
6. Select the datastore for the VM disk, and click Next.
7. Click Next.
8. Set Guest OS Family to Linux and Guest OS Version to Ubuntu Linux (64-bit).
9. Set the following VM hardware configurations:
   - CPU: 4
   - Memory: 8 GB
   - New hard disk: 40 GB
   - New CD/DVD Drive: set to Datastore ISO file, select the Ubuntu Server 22.04 install ISO, and toggle Connect At Power On.
10. Click Next.
11. Click Finish.

12. Right-click the new VM, and click Power → Power On.
13. On the new VM Summary page in vCenter, click Launch Web Console.
14. Select Try or Install Ubuntu Server, and press Enter.
15. Select English, and press Enter.
16. Select Continue without updating, and press Enter.
17. Press Enter.
18. Select Ubuntu Server, and press Enter.
19. After vCenter detects the attached network interface, press Enter.
20. Press Enter.
21. Once the mirror test completes, press Enter.
22. Select Done, and press Enter.
23. Press Enter.
24. Select Continue, and press Enter.
25. Input desired user display name; server hostname; username; and password, select Done, and press Enter. We used PT User, gold-ubuntu-pg16, ptuser/Password1.
26. Press Enter.
27. To toggle Install OpenSSH server, press Enter, select Done, and press Enter.
28. Select Done, and press Enter.
29. Once the OS has installed, select Reboot Now, and press Enter.
30. When prompted, disconnect the OS install ISO in vCenter by clicking the CD/DVD drive in VM Hardware and clicking Disconnect.
31. Reconnect to the remote client, and run the following commands:

```
sudo apt update
sudo apt -y full-upgrade
sudo apt -y install open-vm-tools ca-certificates curl gnupg lsb-release jq sysstat net-tools unzip
chrony lvm2 cloud-guest-utils
sudo systemctl enable --now open-vm-tools
sudo systemctl enable --now chrony
sudo timedatectl set-timezone UTC
```

32. Verify that open-vm-tools and chrony are running, and confirm hostname, time synchronization, and disk:

```
systemctl is-active open-vm-tools
systemctl is-active chrony
timedatectl status
hostnamectl
lsblk
```

33. Add the PostgreSQL 16 package repository and key:

```
curl -fsSL https://www.postgresql.org/media/keys/ACCC4CF8.asc |
  sudo gpg --dearmor -o /usr/share/keyrings/postgresql.gpg

echo "deb [signed-by=/usr/share/keyrings/postgresql.gpg] https://apt.postgresql.org/pub/repos/apt
jammy-pgdg main" |
  sudo tee /etc/apt/sources.list.d/pgdg.list

sudo apt update
```

34. Install PostgreSQL 16:

```
sudo apt -y install postgresql-16 postgresql-client-16 postgresql-contrib-16
```

35. Confirm installation:

```
psql --version
sudo pg_lsclusters
sudo systemctl status postgresql --no-pager
```

36. Edit `/etc/postgresql/16/main/postgresql.conf`, and change `listen_addresses` from `localhost` to `*`
37. Edit `/etc/postgresql/16/main/pg_hba.conf`, and set the IPv4 local connections address field to `0.0.0.0/0`
38. Restart and validate:

```
sudo systemctl restart postgresql
sudo systemctl status postgresql --no-pager
sudo -u postgres psql -c "SELECT version();"
sudo -u postgres psql -c "SHOW shared_buffers;"
```

39. Clean up and shutdown the VM to prepare it for conversion to template:

```
sudo truncate -s 0 /var/log/wtmp /var/log/btmp
sudo cloud-init clean --logs
sudo rm -f /etc/ssh/ssh_host_*
sudo truncate -s 0 /etc/machine-id
sudo shutdown -h now
```

40. In vCenter, right-click the powered off VM, click Template→Convert to Template, and click Yes when prompted.

Provisioning a new PostgreSQL database via vCenter manually

1. Log into vCenter, and start the timer.
2. Right-click the cluster, and click New Virtual Machine.
3. Select Deploy from template, and click Next.
4. Select the VM previously created template, and click Next.
5. Specify a VM name (e.g., provision-test-001), and click Next.
6. Select the destination compute resource, and click Next.
7. Select the destination storage, and click Next.
8. After creation, toggle Power on virtual machine, and click Next.
9. Click Finish.
10. After the VM has powered on automatically, select the VM, click Launch Web Console, and input admin credentials when prompted.
11. Generate new SSH key:

```
sudo ssh-keygen -A
```

12. Create a run directory for sshd:

```
sudo mkdir -p /run/sshd
```

13. Set permissions:

```
sudo chmod 755 /run/sshd
```

14. Validate sshd:

```
sudo sshd -t
```

15. Restart SSH service:

```
sudo systemctl restart ssh
```

16. Verify SSH service:

```
sudo systemctl status ssh --no-pager
```

17. Update the hostname:

```
sudo hostnamectl set-hostname provision-test-001
```

18. Edit /etc/hosts to map 127.0.1.1 to the new hostname:

```
127.0.1.1 provision-test-001
```

19. Refresh package list:

```
sudo apt update
```

20. Apply any available updates:

```
sudo apt -y full-upgrade
```

21. Verify PostgreSQL cluster status:

```
pg_lsclusters
```

22. Verify PostgreSQL service status:

```
sudo systemctl status postgresql --no-pager
```

23. Create a database admin role:

```
sudo -u postgres psql -c "CREATE ROLE dbadmin WITH LOGIN SUPERUSER PASSWORD '<password>';"
```

24. Create a database app user role:

```
sudo -u postgres psql -c "CREATE ROLE appuser WITH LOGIN PASSWORD '<password>';"
```

25. Create a new database:

```
sudo -u postgres createdb -O appuser appdb
```

26. Create schema:

```
sudo -u postgres psql -d appdb -c "CREATE SCHEMA IF NOT EXISTS app AUTHORIZATION appuser;"
```

27. Validate service readiness:

```
pg_isready -h localhost -p 5432
```

28. Validate database creation:

```
sudo -u postgres psql -d appdb -c "SELECT current_database();"
```

Cloning a database manually

1. Log into vCenter, and start the timer.
2. Right click the cluster, and click New Virtual Machine.
3. Select Deploy from template, and click Next.
4. Select the VM previously created template, and click Next.
5. Specify a VM name (e.g., provision-test-001), and click Next.
6. Select the destination compute resource, and click Next.
7. Select the destination storage, and click Next.
8. After creation, toggle Power on virtual machine, and click Next.
9. Click Finish.
10. While vCenter deploys the new target clone VM, select the source database VM created in the manual provisioning scenario, click Launch Web Console, and input admin credentials when prompted.
11. Create a directory for the source database export:

```
sudo -u postgres mkdir -p /var/lib/postgresql/pgclone
```

12. Create the database export:

```
sudo -u postgres pg_dump -d appdb -Fc -f /var/lib/postgresql/pgclone/source-db.dump
```

13. Verify the database dump worked:

```
sudo -u postgres ls -lh /var/lib/postgresql/pgclone/source-db.dump
```

14. Return to vCenter once the clone VM has finished deploying, select it, click Launch Web Console, and input admin credentials.
15. Generate new SSH key:

```
sudo ssh-keygen -A
```

16. Create a run directory for sshd:

```
sudo mkdir -p /run/sshd
```

17. Set permissions:

```
sudo chmod 755 /run/sshd
```

18. Validate sshd:

```
sudo sshd -t
```

19. Restart SSH service:

```
sudo systemctl restart ssh
```

20. Verify SSH service:

```
sudo systemctl status ssh --no-pager
```

21. Set the target VM's hostname:

```
sudo hostnamectl set-hostname clone-test-001
```

22. Update /etc/hosts with new hostname:

```
127.0.1.1 clone-test-001
```

23. Refresh package list:

```
sudo apt update
```

24. Apply any available updates:

```
sudo apt -y full-upgrade
```

25. Verify PostgreSQL cluster status:

```
sudo pg_listclusters
```

26. Verify PostgreSQL service status:

```
sudo systemctl status postgresql --no-pager
```

27. Create a database admin role:

```
sudo -u postgres psql -c "CREATE ROLE dbadmin WITH LOGIN SUPERUSER PASSWORD '<password>';"
```

28. Create a database app user role:

```
sudo -u postgres psql -c "CREATE ROLE appuser WITH LOGIN PASSWORD '<password>';"
```

29. Create a new database:

```
sudo -u postgres createdb -O appuser appdb-clone
```

30. Create a directory for the database import:

```
sudo -u postgres mkdir -p /var/lib/postgresql/pgclone
```

31. Switch to the web console for the source VM, and copy the database export to the newly created clone target VM:

```
sudo scp /var/lib/postgresql/pgclone/source-db.dump ptuser@<target clone VM IP>:/tmp/source-db.dump
```

32. Switch back to the target clone VM web console, and move the copied database dump:

```
sudo mv /tmp/source-db.dump /var/lib/postgresql/pgclone/source-db.dump
```

33. Set permissions:

```
sudo chown postgres:postgres /var/lib/postgresql/pgclone/source-db.dump
```

34. Validate service readiness:

```
pg_isready -h localhost -p 5432
```

35. Validate database creation:

```
sudo -u postgres psql -d appdb-clone -c "SELECT current_database();"
```

36. Restore the database:

```
sudo -u postgres pg_restore -d appdb-clone --no-owner --role=appuser /var/lib/postgresql/
pgclone/source-db.dump
```

37. Refresh optimizer statistics:

```
sudo -u postgres psql -d appdb-clone -c "ANALYZE;"
```

38. Validate restored database:

```
sudo -u postgres psql -d appdb-clone -c "SELECT count(*) FROM pg_tables WHERE schemaname NOT IN
('pg_catalog','information_schema');"
```

Upgrading from PostgreSQL 16 to 17 manually

1. Log into vCenter, and start the timer.
2. Select the VM created in the first scenario, click Launch Web Console, and enter credentials when prompted.
3. Confirm current PostgreSQL version:

```
sudo -u postgres psql -c "SELECT version();"
```

4. Confirm PostgreSQL 16 service is running:

```
sudo systemctl status postgresql --no-pager
```

5. List database inventory:

```
sudo -u postgres psql -c "SELECT datname FROM pg_database WHERE datistemplate = false;"
```

6. Create rollback backup folder:

```
sudo -u postgres mkdir -p /var/lib/postgresql/pgupgrade
```

7. Capture global PostgreSQL objects:

```
sudo -u postgres pg_dumpall --globals-only -f /var/lib/postgresql/pgupgrade/globals.sql
```

8. Backup existing database:

```
sudo -u postgres pg_dump -d appdb -Fc -f /var/lib/postgresql/pgupgrade/appdb-preupgrade.dump
```

9. Refresh package list:

```
sudo apt update
```

10. Install PostgreSQL 17 packages:

```
sudo apt -y install postgresql-17 postgresql-client-17 postgresql-contrib-17
```

11. Verify cluster state after package installation:

```
sudo pg_lsclusters
```

12. Upgrade the cluster:

```
sudo pg_upgradecluster -v 17 16 main
```

13. Verify post-upgrade cluster state:

```
sudo pg_lsclusters
```

14. Refresh statistics:

```
sudo -u postgres vacuumdb --all --analyze-in-stages
```

15. Validate PostgreSQL version:

```
sudo -u postgres psql -c "SELECT version();"
```

16. Validate cluster readiness:

```
pg_isready -h localhost -p 5432
```

17. Validate the appdb database:

```
sudo -u postgres psql -d appdb -c "SELECT now();"
```

18. Review logs to check for any errors during upgrade process:

```
sudo journalctl -u postgresql -n 100 --no-pager
```

19. Drop the old PostgreSQL 16 cluster:

```
sudo pg_dropcluster 16 main --stop
```

Provisioning a new three-node PostgreSQL 16 HA cluster in DSM

1. Log into DSM, and start the timer.
2. Click Databases → Postgres.
3. Click Create Database.
4. Set Version to 16.11+vmware.v9.1.0.0.
5. Input the desired instance name and admin password.
6. Click Next.
7. Select Single vSphere Cluster HA.
8. Click Next.
9. Set the VM Class to medium.
10. Set the Disk Size to 40GB.
11. Click Next.
12. Click Next.
13. Click Create. Stop the timer when the database status changes from In Progress to Ready in the Postgres databases listing, and record the result.

Scaling out an existing single-node PostgreSQL database to a three-node HA cluster in DSM

1. Log into DSM, and start the timer.
2. Click Databases → Postgres.
3. Click the instance name of the previously created single-node database.
4. Next to Data Availability and Remote Replication, click Edit.
5. From the Topology drop-down menu, select Single vSphere Cluster HA.
6. Click Save. When the DSM UI reports the scale-out complete and the database status returns to Ready, stop the timer and record the result.

Provisioning a new three-node PostgreSQL HA cluster via vCenter manually

1. Log into vCenter, and start the timer.
2. Right-click the cluster, and click New Virtual Machine.
3. Select Deploy from template, and click Next.
4. Select the gold-ubuntu-pg16 template, and click Next.
5. Specify a VM name (pg-primary-001), and click Next.
6. Select the destination compute resource (first ESXi host in the cluster), and click Next.
7. Select the destination storage, and click Next.
8. After creation, toggle Power on virtual machine, and click Next.
9. Click Finish.
10. Right-click the cluster, and click New Virtual Machine.
11. Select Deploy from template, and click Next.
12. Select the gold-ubuntu-pg16 template, and click Next.
13. Specify a VM name (pg-replica-001), and click Next.
14. Select the destination compute resource (second ESXi host in the cluster, different from pg-primary-001), and click Next.
15. Select the destination storage, and click Next.
16. After creation, toggle Power on virtual machine, and click Next.
17. Click Finish.
18. Right-click the cluster, and click New Virtual Machine.
19. Select Deploy from template, and click Next.
20. Select the gold-ubuntu-pg16 template, and click Next.
21. Specify a VM name (pg-witness-001), and click Next.
22. Select the destination compute resource (third ESXi host in the cluster, different from pg-primary-001 and pg-replica-001), and click Next.
23. Select the destination storage, and click Next.
24. After creation, toggle Power on virtual machine, and click Next.
25. Click Finish.
26. After the VMs have powered on automatically, select pg-primary-001, click Launch Web Console, and input admin credentials when prompted.

27. Regenerate the machine ID so the node receives a unique DHCP lease (clones from the gold template otherwise share an identifier and are assigned the same address):

```
sudo truncate -s 0 /etc/machine-id
sudo rm -f /var/lib/dbus/machine-id
sudo systemd-machine-id-setup
sudo ln -sf /etc/machine-id /var/lib/dbus/machine-id
```

28. Generate new SSH key:

```
sudo ssh-keygen -A
```

29. Create a run directory for sshd:

```
sudo mkdir -p /run/sshd
```

30. Set permissions:

```
sudo chmod 755 /run/sshd
```

31. Validate sshd:

```
sudo sshd -t
```

32. Restart SSH service:

```
sudo systemctl restart ssh
```

33. Verify SSH service:

```
sudo systemctl status ssh --no-pager
```

34. Reboot the node to acquire a unique DHCP lease with the new machine ID:

```
sudo reboot
```

35. Select pg-replica-001, click Launch Web Console, input admin credentials.
36. Regenerate the machine ID so the node receives a unique DHCP lease (clones from the gold template otherwise share an identifier and are assigned the same address):

```
sudo truncate -s 0 /etc/machine-id
sudo rm -f /var/lib/dbus/machine-id
sudo systemd-machine-id-setup
sudo ln -sf /etc/machine-id /var/lib/dbus/machine-id
```

37. Generate new SSH key:

```
sudo ssh-keygen -A
```

38. Create a run directory for sshd:

```
sudo mkdir -p /run/sshd
```

39. Set permissions:

```
sudo chmod 755 /run/sshd
```

40. Validate sshd:

```
sudo sshd -t
```

41. Restart SSH service:

```
sudo systemctl restart ssh
```

42. Verify SSH service:

```
sudo systemctl status ssh --no-pager
```

43. Reboot the node to acquire a unique DHCP lease with the new machine ID:

```
sudo reboot
```

44. Select pg-witness-001, click Launch Web Console, input admin credentials.
45. Regenerate the machine ID so the node receives a unique DHCP lease (clones from the gold template otherwise share an identifier and are assigned the same address):

```
sudo truncate -s 0 /etc/machine-id
sudo rm -f /var/lib/dbus/machine-id
sudo systemd-machine-id-setup
sudo ln -sf /etc/machine-id /var/lib/dbus/machine-id
```

46. Generate new SSH key:

```
sudo ssh-keygen -A
```

47. Create a run directory for sshd:

```
sudo mkdir -p /run/sshd
```

48. Set permissions:

```
sudo chmod 755 /run/sshd
```

49. Validate sshd:

```
sudo sshd -t
```

50. Restart SSH service:

```
sudo systemctl restart ssh
```

51. Verify SSH service:

```
sudo systemctl status ssh --no-pager
```

52. Reboot the node to acquire a unique DHCP lease with the new machine ID:

```
sudo reboot
```

53. After all three nodes have rebooted, confirm in the vCenter Summary tab that each node now shows a unique IPv4 address. SSH is now available on all three nodes. Close the web console. Perform all remaining steps over SSH, connecting to each node by IP as ptuser (for example, ssh ptuser@<pg-primary-IP>). Open a separate SSH session to each node as needed.
54. Open an SSH session to pg-primary-001, and update the hostname:

```
sudo hostnamectl set-hostname pg-primary-001
```

55. Edit /etc/hosts to map 127.0.1.1 to the new hostname, and add static entries for the other two cluster members:

```
# On pg-primary-001, open the hosts file with a text editor:
sudo nano /etc/hosts
# Set the 127.0.1.1 line to the local hostname, and add the three node entries:
127.0.1.1        pg-primary-001
<pg-primary-IP>  pg-primary-001
<pg-replica-IP>  pg-replica-001
<pg-witness-IP>  pg-witness-001
```

56. Refresh the package list:

```
sudo apt update
```

57. Apply any available updates:

```
sudo apt -y full-upgrade
```

58. Install the repmgr package:

```
sudo apt -y install postgresql-16-repmgr
dpkg -l | grep repmgr
```

59. Verify PostgreSQL cluster status:

```
sudo pg_lsclusters
```

60. Verify PostgreSQL service status:

```
sudo systemctl status postgresql --no-pager
```

61. Create a database admin role:

```
sudo -u postgres psql -c "CREATE ROLE dbadmin WITH LOGIN SUPERUSER PASSWORD '<password>';"
```

62. Create a database app user role:

```
sudo -u postgres psql -c "CREATE ROLE appuser WITH LOGIN PASSWORD '<password>';"
```

63. Create a new database:

```
sudo -u postgres createdb -O appuser appdb
```

64. Create the schema:

```
sudo -u postgres psql -d appdb -c "CREATE SCHEMA IF NOT EXISTS app AUTHORIZATION appuser;"
```

65. Create the repmgr role and metadata database:

```
sudo -u postgres psql -c "CREATE ROLE repmgr WITH SUPERUSER LOGIN PASSWORD '<password>';"
sudo -u postgres createdb repmgr -O repmgr
```

66. Create a .pgpass file for the postgres user so repmgr can authenticate to the other nodes non-interactively:

```
sudo -u postgres bash -c 'echo "*:5432:repmgr:repmgr:<password>" > ~/.pgpass'
sudo -u postgres bash -c 'chmod 600 ~/.pgpass'
```

67. Edit /etc/postgresql/16/main/postgresql.conf, and set the following parameters:

```
listen_addresses = '*'
wal_level = replica
max_wal_senders = 10
max_replication_slots = 10
hot_standby = on
wal_log_hints = on
shared_preload_libraries = 'repmgr'
archive_mode = on
archive_command = '/bin/true'
```

68. Edit /etc/postgresql/16/main/pg_hba.conf, and add entries for replication and repmgr connections from all three cluster members:

```
host  replication  repmgr      <pg-primary-IP>/32  scram-sha-256
host  replication  repmgr      <pg-replica-IP>/32  scram-sha-256
host  replication  repmgr      <pg-witness-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-primary-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-replica-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-witness-IP>/32  scram-sha-256
```

69. Restart PostgreSQL and validate:

```
sudo systemctl restart postgresql
sudo systemctl status postgresql --no-pager
```

70. Validate service readiness:

```
pg_isready -h localhost -p 5432
```

71. Validate database creation:

```
sudo -u postgres psql -d appdb -c "SELECT current_database();"
```

72. Create /etc/repmgr.conf on pg-primary-001:

```
node_id=1
node_name='pg-primary-001'
conninfo='host=<pg-primary-IP> user=repmgr dbname=repmgr password=<password>'
data_directory='/var/lib/postgresql/16/main'
failover='automatic'
promote_command='/usr/bin/repmgr standby promote -f /etc/repmgr.conf --log-to-file'
follow_command='/usr/bin/repmgr standby follow -f /etc/repmgr.conf --log-to-file
--upstream-node-id=%n'
monitoring_history=yes
log_file='/var/log/postgresql/repmgr.log'
```

73. Register pg-primary-001 with repmgr:

```
sudo -u postgres repmgr -f /etc/repmgr.conf primary register
sudo -u postgres repmgr -f /etc/repmgr.conf cluster show
```

74. Configure and start the repmgrd daemon on pg-primary-001:

```
# Set the following in /etc/default/repmgrd:
REPMGRD_ENABLED=yes
REPMGRD_CONF="/etc/repmgr.conf"
sudo systemctl enable repmgrd
sudo systemctl restart repmgrd
sleep 2
pgrep -af repmgrd
```

75. Open an SSH session to pg-replica-001, and update the hostname:

```
sudo hostnamectl set-hostname pg-replica-001
```

76. Edit /etc/hosts to map 127.0.1.1 to the new hostname, and add static entries for the other two cluster members:

```
# On pg-replica-001, open the hosts file with a text editor:
sudo nano /etc/hosts
# Set the 127.0.1.1 line to the local hostname, and add the three node entries:
127.0.1.1        pg-replica-001
<pg-primary-IP>  pg-primary-001
<pg-replica-IP>  pg-replica-001
<pg-witness-IP>  pg-witness-001
```

77. Refresh package list:

```
sudo apt update
```

78. Apply any available updates:

```
sudo apt -y full-upgrade
```

79. Install the repmgr package:

```
sudo apt -y install postgresql-16-repmgr
dpkg -l | grep repmgr
```

80. Edit /etc/postgresql/16/main/postgresql.conf, and set the following parameters:

```
listen_addresses = '*'
wal_level = replica
max_wal_senders = 10
max_replication_slots = 10
hot_standby = on
wal_log_hints = on
shared_preload_libraries = 'repmgr'
archive_mode = on
archive_command = '/bin/true'
```

81. Edit /etc/postgresql/16/main/pg_hba.conf, and add entries for replication and repmgr connections from all three cluster members:

```
host  replication  repmgr      <pg-primary-IP>/32  scram-sha-256
host  replication  repmgr      <pg-replica-IP>/32  scram-sha-256
host  replication  repmgr      <pg-witness-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-primary-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-replica-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-witness-IP>/32  scram-sha-256
```

82. Create a .pgpass file for the postgres user so repmgr can authenticate to the other nodes non-interactively:

```
sudo -u postgres bash -c 'echo "*:5432:repmgr:repmgr:<password>" > ~/.pgpass'
sudo -u postgres bash -c 'chmod 600 ~/.pgpass'
```

83. Create /etc/repmgr.conf on pg-replica-001:

```
node_id=2
node_name='pg-replica-001'
conninfo='host=<pg-replica-IP> user=repmgr dbname=repmgr password=<password>'
data_directory='/var/lib/postgresql/16/main'
failover='automatic'
promote_command='/usr/bin/repmgr standby promote -f /etc/repmgr.conf --log-to-file'
follow_command='/usr/bin/repmgr standby follow -f /etc/repmgr.conf --log-to-file
--upstream-node-id=%n'
monitoring_history=yes
log_file='/var/log/postgresql/repmgr.log'
```

84. Stop PostgreSQL and clear the local data directory:

```
sudo systemctl stop postgresql
sudo -u postgres bash -c 'rm -rf /var/lib/postgresql/16/main/*'
```

85. Clone the primary (dry run first, then the actual clone):

```
sudo -u postgres repmgr -h <pg-primary-IP> -U repmgr -d repmgr \
  -f /etc/repmgr.conf standby clone --dry-run
sudo -u postgres repmgr -h <pg-primary-IP> -U repmgr -d repmgr \
  -f /etc/repmgr.conf standby clone
```

86. Start PostgreSQL and verify recovery mode (expect t):

```
sudo systemctl start postgresql
sudo -u postgres psql -c "SELECT pg_is_in_recovery();"
```

87. Register pg-replica-001 with repmgr:

```
sudo -u postgres repmgr -f /etc/repmgr.conf standby register
sudo -u postgres repmgr -f /etc/repmgr.conf cluster show
```

88. Configure and start the repmgrd daemon on pg-replica-001:

```
# Set the following in /etc/default/repmgrd:
REPMGRD_ENABLED=yes
REPMGRD_CONF="/etc/repmgr.conf"
sudo systemctl enable repmgrd
sudo systemctl restart repmgrd
sleep 2
pgrep -af repmgrd
```

89. Open an SSH session to pg-witness-001, and update the hostname:

```
sudo hostnamectl set-hostname pg-witness-001
```

90. Edit /etc/hosts to map 127.0.1.1 to the new hostname, and add static entries for the other two cluster members:

```
# On pg-witness-001, open the hosts file with a text editor:
sudo nano /etc/hosts
# Set the 127.0.1.1 line to the local hostname, and add the three node entries:
127.0.1.1        pg-witness-001
<pg-primary-IP>  pg-primary-001
<pg-replica-IP>  pg-replica-001
<pg-witness-IP>  pg-witness-001
```

91. Refresh package list:

```
sudo apt update
```

92. Apply any available updates:

```
sudo apt -y full-upgrade
```

93. Install the repmgr package:

```
sudo apt -y install postgresql-16-repmgr
dpkg -l | grep repmgr
```

94. Re-initialize the witness PostgreSQL cluster so that it has a unique database system identifier:

```
sudo pg_dropcluster --stop 16 main
sudo pg_createcluster --start 16 main
```

95. Create the repmgr role and metadata database on pg-witness-001:

```
sudo -u postgres psql -c "CREATE ROLE repmgr WITH SUPERUSER LOGIN PASSWORD '<password>';"
sudo -u postgres createdb repmgr -O repmgr
```

96. Create a .pgpass file for the postgres user so repmgr can authenticate to the other nodes non-interactively:

```
sudo -u postgres bash -c 'echo "*:5432:repmgr:repmgr:<password>" > ~/.pgpass'
sudo -u postgres bash -c 'chmod 600 ~/.pgpass'
```

97. Edit /etc/postgresql/16/main/postgresql.conf on pg-witness-001, and set the following parameters:

```
listen_addresses = '*'
shared_preload_libraries = 'repmgr'
```

98. Edit /etc/postgresql/16/main/pg_hba.conf on pg-witness-001, and add entries for repmgr connections from all three cluster members:

```
host  repmgr  repmgr  <pg-primary-IP>/32  scram-sha-256
host  repmgr  repmgr  <pg-replica-IP>/32  scram-sha-256
host  repmgr  repmgr  <pg-witness-IP>/32  scram-sha-256
```

99. Restart PostgreSQL on pg-witness-001:

```
sudo systemctl restart postgresql
```

100. Create /etc/repmgr.conf on pg-witness-001:

```
node_id=3
node_name='pg-witness-001'
conninfo='host=<pg-witness-IP> user=repmgr dbname=repmgr password=<password>'
data_directory='/var/lib/postgresql/16/main'
failover='automatic'
promote_command='/usr/bin/repmgr standby promote -f /etc/repmgr.conf --log-to-file'
follow_command='/usr/bin/repmgr standby follow -f /etc/repmgr.conf --log-to-file
--upstream-node-id=%n'
monitoring_history=yes
log_file='/var/log/postgresql/repmgr.log'
```

101. Register pg-witness-001 with repmgr:

```
sudo -u postgres repmgr -f /etc/repmgr.conf witness register -h <pg-primary-IP>
sudo -u postgres repmgr -f /etc/repmgr.conf cluster show
```

102. Configure and start the repmgrd daemon on pg-witness-001:

```
# Set the following in /etc/default/repmgrd:
REPMGRD_ENABLED=yes
REPMGRD_CONF="/etc/repmgr.conf"
sudo systemctl enable repmgrd
sudo systemctl restart repmgrd
sleep 2
pgrep -af repmgrd
```

103. In the SSH session to pg-primary-001, validate the full cluster:

```
sudo -u postgres repmgr -f /etc/repmgr.conf cluster show
sudo -u postgres repmgr -f /etc/repmgr.conf service status
```

104. Run a smoke-test psql to the primary. Stop the timer when repmgr cluster show reports primary running, standby running, witness running; repmgr service status reports repmgrd running on all three nodes; and the smoke-test psql succeeds. Record the result.

```
psql -h <pg-primary-IP> -U appuser -d appdb -c "SELECT 1;"
```

Scaling out an existing single-node PostgreSQL database to a three-node HA cluster manually

1. Log into vCenter, and start the timer.
2. Right-click the cluster, and click New Virtual Machine.
3. Select Deploy from template, and click Next.
4. Select the gold-ubuntu-pg16 template, and click Next.
5. Specify a VM name (pg-scalereplica-001), and click Next.
6. Select the destination compute resource (an ESXi host different from the one hosting pg-scaleprimary-001), and click Next.
7. Select the destination storage, and click Next.
8. After creation, toggle Power on virtual machine, and click Next.
9. Click Finish.
10. Right-click the cluster, and click New Virtual Machine.
11. Select Deploy from template, and click Next.
12. Select the gold-ubuntu-pg16 template, and click Next.
13. Specify a VM name (pg-scalewitness-001), and click Next.
14. Select the destination compute resource (the third ESXi host, different from pg-scaleprimary-001 and pg-scalereplica-001), and click Next.
15. Select the destination storage, and click Next.
16. After creation, toggle Power on virtual machine, and click Next.
17. Click Finish.
18. Once pg-scalereplica-001 has powered on, select it, click Launch Web Console, input admin credentials.
19. Regenerate the machine ID so the node receives a unique DHCP lease (clones from the gold template otherwise share an identifier and are assigned the same address):

```
sudo truncate -s 0 /etc/machine-id
sudo rm -f /var/lib/dbus/machine-id
sudo systemd-machine-id-setup
sudo ln -sf /etc/machine-id /var/lib/dbus/machine-id
```

20. Generate new SSH key:

```
sudo ssh-keygen -A
```

21. Create a run directory for sshd:

```
sudo mkdir -p /run/sshd
```

22. Set permissions:

```
sudo chmod 755 /run/sshd
```

23. Validate sshd:

```
sudo sshd -t
```

24. Restart SSH service:

```
sudo systemctl restart ssh
```

25. Verify SSH service:

```
sudo systemctl status ssh --no-pager
```

26. Reboot the node to acquire a unique DHCP lease with the new machine ID:

```
sudo reboot
```

27. Select pg-scalewitness-001, click Launch Web Console, input admin credentials.
28. Regenerate the machine ID so the node receives a unique DHCP lease (clones from the gold template otherwise share an identifier and are assigned the same address):

```
sudo truncate -s 0 /etc/machine-id
sudo rm -f /var/lib/dbus/machine-id
sudo systemd-machine-id-setup
sudo ln -sf /etc/machine-id /var/lib/dbus/machine-id
```

29. Generate new SSH key:

```
sudo ssh-keygen -A
```

30. Create a run directory for sshd:

```
sudo mkdir -p /run/sshd
```

31. Set permissions:

```
sudo chmod 755 /run/sshd
```

32. Validate sshd:

```
sudo sshd -t
```

33. Restart SSH service:

```
sudo systemctl restart ssh
```

34. Verify SSH service:

```
sudo systemctl status ssh --no-pager
```

35. Reboot the node to acquire a unique DHCP lease with the new machine ID:

```
sudo reboot
```

36. After the two new nodes have rebooted, confirm in the vCenter Summary tab that pg-scalereplica-001 and pg-scalewitness-001 each show a unique IPv4 address (distinct from pg-scaleprimary-001 and from each other).
37. Open an SSH session to the existing primary (by IP, as <pg-scaleprimary-IP>), and update the hostname from its original value (provision-test-001) to the cluster name:

```
sudo hostnamectl set-hostname pg-scaleprimary-001
```

38. Edit /etc/hosts to map 127.0.1.1 to the new hostname, and add static entries for the other two cluster members (substituting the actual IPs):

```
# On the existing primary, open the hosts file:
sudo nano /etc/hosts
# Set the 127.0.1.1 line to the new hostname, and add the three node entries:
127.0.1.1               pg-scaleprimary-001
<pg-scaleprimary-IP>  pg-scaleprimary-001
<pg-scalereplica-IP>          pg-scalereplica-001
<pg-scalewitness-IP>          pg-scalewitness-001
```

39. Install the repmgr package on the existing primary:

```
sudo apt update
sudo apt -y install postgresql-16-repmgr
dpkg -l | grep repmgr
```

40. Create the repmgr role and metadata database on the existing primary:

```
sudo -u postgres psql -c "CREATE ROLE repmgr WITH SUPERUSER LOGIN PASSWORD '<password>';"
sudo -u postgres createdb repmgr -O repmgr
```

41. Create a .pgpass file for the postgres user so repmgr can authenticate to the other nodes non-interactively:

```
sudo -u postgres bash -c 'echo "*:5432:repmgr:repmgr:<password>" > ~/.pgpass'
sudo -u postgres bash -c 'chmod 600 ~/.pgpass'
```

42. Edit /etc/postgresql/16/main/postgresql.conf on the existing primary, and set the following parameters:

```
listen_addresses = '*'
wal_level = replica
max_wal_senders = 10
max_replication_slots = 10
hot_standby = on
wal_log_hints = on
shared_preload_libraries = 'repmgr'
archive_mode = on
archive_command = '/bin/true'
```

43. Edit /etc/postgresql/16/main/pg_hba.conf on the existing primary, and add entries for replication and repmgr connections from all three cluster members:

```
host  replication  repmgr      <pg-scaleprimary-IP>/32  scram-sha-256
host  replication  repmgr      <pg-scalereplica-IP>/32  scram-sha-256
host  replication  repmgr      <pg-scalewitness-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-scaleprimary-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-scalereplica-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-scalewitness-IP>/32  scram-sha-256
```

44. Restart PostgreSQL on the existing primary:

```
sudo systemctl restart postgresql
sudo systemctl status postgresql --no-pager
```

45. Create /etc/repmgr.conf on the existing primary:

```
node_id=1
node_name='pg-scaleprimary-001'
conninfo='host=<pg-scaleprimary-IP> user=repmgr dbname=repmgr password=<password>'
data_directory='/var/lib/postgresql/16/main'
failover='automatic'
promote_command='/usr/bin/repmgr standby promote -f /etc/repmgr.conf --log-to-file'
follow_command='/usr/bin/repmgr standby follow -f /etc/repmgr.conf --log-to-file
--upstream-node-id=%n'
monitoring_history=yes
log_file='/var/log/postgresql/repmgr.log'
```

46. Register the existing primary with repmgr:

```
sudo -u postgres repmgr -f /etc/repmgr.conf primary register
sudo -u postgres repmgr -f /etc/repmgr.conf cluster show
```

47. Configure and start the repmgrd daemon on the existing primary:

```
# Set the following in /etc/default/repmgrd:
REPMGRD_ENABLED=yes
REPMGRD_CONF="/etc/repmgr.conf"
sudo systemctl enable repmgrd
sudo systemctl restart repmgrd
sleep 2
pgrep -af repmgrd
```

48. Open an SSH session to pg-scalereplica-001, and update the hostname:

```
sudo hostnamectl set-hostname pg-scalereplica-001
```

49. Edit /etc/hosts to map 127.0.1.1 to the new hostname, and add static entries for the other two cluster members:

```
# On pg-scalereplica-001, open the hosts file with a text editor:
sudo nano /etc/hosts
# Set the 127.0.1.1 line to the local hostname, and add the three node entries:
127.0.1.1               pg-scalereplica-001
<pg-scaleprimary-IP>  pg-scaleprimary-001
<pg-scalereplica-IP>          pg-scalereplica-001
<pg-scalewitness-IP>          pg-scalewitness-001
```

50. Refresh package list:

```
sudo apt update
```

51. Apply any available updates:

```
sudo apt -y full-upgrade
```

52. Install the repmgr package:

```
sudo apt -y install postgresql-16-repmgr
dpkg -l | grep repmgr
```

53. Edit /etc/postgresql/16/main/postgresql.conf, and set the following parameters (identical to the primary, so the replica is promotable):

```
listen_addresses = '*'
wal_level = replica
max_wal_senders = 10
max_replication_slots = 10
hot_standby = on
wal_log_hints = on
shared_preload_libraries = 'repmgr'
archive_mode = on
archive_command = '/bin/true'
```

54. Edit /etc/postgresql/16/main/pg_hba.conf, and add entries for replication and repmgr connections from all three cluster members (identical to the primary):

```
host  replication  repmgr      <pg-scaleprimary-IP>/32  scram-sha-256
host  replication  repmgr      <pg-scalereplica-IP>/32  scram-sha-256
host  replication  repmgr      <pg-scalewitness-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-scaleprimary-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-scalereplica-IP>/32  scram-sha-256
host  repmgr       repmgr      <pg-scalewitness-IP>/32  scram-sha-256
```

55. Create a .pgpass file for the postgres user so repmgr can authenticate to the other nodes non-interactively:

```
sudo -u postgres bash -c 'echo "*:5432:repmgr:repmgr:<password>" > ~/.pgpass'
sudo -u postgres bash -c 'chmod 600 ~/.pgpass'
```

56. Create /etc/repmgr.conf on pg-scalereplica-001 (before cloning):

```
node_id=2
node_name='pg-scalereplica-001'
conninfo='host=<pg-scalereplica-IP> user=repmgr dbname=repmgr password=<password>'
data_directory='/var/lib/postgresql/16/main'
failover='automatic'
promote_command='/usr/bin/repmgr standby promote -f /etc/repmgr.conf --log-to-file'
follow_command='/usr/bin/repmgr standby follow -f /etc/repmgr.conf --log-to-file
--upstream-node-id=%n'
monitoring_history=yes
log_file='/var/log/postgresql/repmgr.log'
```

57. Stop PostgreSQL and clear the local data directory:

```
sudo systemctl stop postgresql
sudo -u postgres bash -c 'rm -rf /var/lib/postgresql/16/main/*'
```

58. Clone the primary (dry run first, then the actual clone):

```
sudo -u postgres repmgr -h <pg-scaleprimary-IP> -U repmgr -d repmgr \
  -f /etc/repmgr.conf standby clone --dry-run
sudo -u postgres repmgr -h <pg-scaleprimary-IP> -U repmgr -d repmgr \
  -f /etc/repmgr.conf standby clone
```

59. Start PostgreSQL and verify recovery mode (expect t):

```
sudo systemctl start postgresql
sudo -u postgres psql -c "SELECT pg_is_in_recovery();"
```

60. Register pg-scalereplica-001 with repmgr:

```
sudo -u postgres repmgr -f /etc/repmgr.conf standby register
sudo -u postgres repmgr -f /etc/repmgr.conf cluster show
```

61. Configure and start the repmgrd daemon on pg-scalereplica-001:

```
# Set the following in /etc/default/repmgrd:
REPMGRD_ENABLED=yes
REPMGRD_CONF="/etc/repmgr.conf"
sudo systemctl enable repmgrd
sudo systemctl restart repmgrd
sleep 2
pgrep -af repmgrd
```

62. Open an SSH session to pg-scalewitness-001, and update the hostname:

```
sudo hostnamectl set-hostname pg-scalewitness-001
```

63. Edit /etc/hosts to map 127.0.1.1 to the new hostname, and add static entries for the other two cluster members:

```
# On pg-scalewitness-001, open the hosts file with a text editor:
sudo nano /etc/hosts
# Set the 127.0.1.1 line to the local hostname, and add the three node entries:
127.0.1.1               pg-scalewitness-001
<pg-scaleprimary-IP>  pg-scaleprimary-001
<pg-scalereplica-IP>          pg-scalereplica-001
<pg-scalewitness-IP>          pg-scalewitness-001
```

64. Refresh package list:

```
sudo apt update
```

65. Apply any available updates:

```
sudo apt -y full-upgrade
```

66. Install the repmgr package:

```
sudo apt -y install postgresql-16-repmgr
dpkg -l | grep repmgr
```

67. Re-initialize the witness PostgreSQL cluster:

```
sudo pg_dropcluster --stop 16 main
sudo pg_createcluster --start 16 main
```

68. Create the repmgr role and metadata database on pg-scalewitness-001:

```
sudo -u postgres psql -c "CREATE ROLE repmgr WITH SUPERUSER LOGIN PASSWORD '<password>';"
sudo -u postgres createdb repmgr -O repmgr
```

69. Create a .pgpass file for the postgres user so repmgr can authenticate to the other nodes non-interactively:

```
sudo -u postgres bash -c 'echo "*:5432:repmgr:repmgr:<password>" > ~/.pgpass'
sudo -u postgres bash -c 'chmod 600 ~/.pgpass'
```

70. Edit /etc/postgresql/16/main/postgresql.conf on pg-scalewitness-001, and set the following parameters:

```
listen_addresses = '*'
shared_preload_libraries = 'repmgr'
```

71. Edit /etc/postgresql/16/main/pg_hba.conf on pg-scalewitness-001, and add entries for repmgr connections from all three cluster members:

```
host  repmgr  repmgr  <pg-scaleprimary-IP>/32  scram-sha-256
host  repmgr  repmgr  <pg-scalereplica-IP>/32  scram-sha-256
host  repmgr  repmgr  <pg-scalewitness-IP>/32  scram-sha-256
```

72. Restart PostgreSQL on pg-scalewitness-001:

```
sudo systemctl restart postgresql
```

73. Create /etc/repmgr.conf on pg-scalewitness-001:

```
node_id=3
node_name='pg-scalewitness-001'
conninfo='host=<pg-scalewitness-IP> user=repmgr dbname=repmgr password=<password>'
data_directory='/var/lib/postgresql/16/main'
failover='automatic'
promote_command='/usr/bin/repmgr standby promote -f /etc/repmgr.conf --log-to-file'
follow_command='/usr/bin/repmgr standby follow -f /etc/repmgr.conf --log-to-file
--upstream-node-id=%n'
monitoring_history=yes
log_file='/var/log/postgresql/repmgr.log'
```

74. Register pg-scalewitness-001 with repmgr:

```
sudo -u postgres repmgr -f /etc/repmgr.conf witness register -h <pg-scaleprimary-IP>
sudo -u postgres repmgr -f /etc/repmgr.conf cluster show
```

75. Configure and start the repmgrd daemon on pg-scalewitness-001:

```
# Set the following in /etc/default/repmgrd:
REPMGRD_ENABLED=yes
REPMGRD_CONF="/etc/repmgr.conf"
sudo systemctl enable repmgrd
sudo systemctl restart repmgrd
sleep 2
pgrep -af repmgrd
```

76. In the SSH session to pg-scaleprimary-001, validate the full cluster:

```
sudo -u postgres repmgr -f /etc/repmgr.conf cluster show
sudo -u postgres repmgr -f /etc/repmgr.conf service status
```

77. Run a smoke-test psql to the primary. Stop the timer when repmgr cluster show reports primary running, standby running, witness running; repmgr service status reports repmgrd running on all three nodes; and the smoke-test psql succeeds. Record the result.

```
psql -h <pg-scaleprimary-IP> -U appuser -d appdb -c "SELECT 1;"
```

This project was commissioned by Broadcom.

Read the report ▶

Primary contributors

Tech: Chris B.

Writing: Nathan P.

Design: Laura K.

PM: Scott Luchene

How we created this report

A PT team, which includes the contributors we've listed and others, created this report and performed the technical work behind it. We used AI to aid in research, methodology development, report outlining, and editing.

Principled Technologies®

Facts matter.®