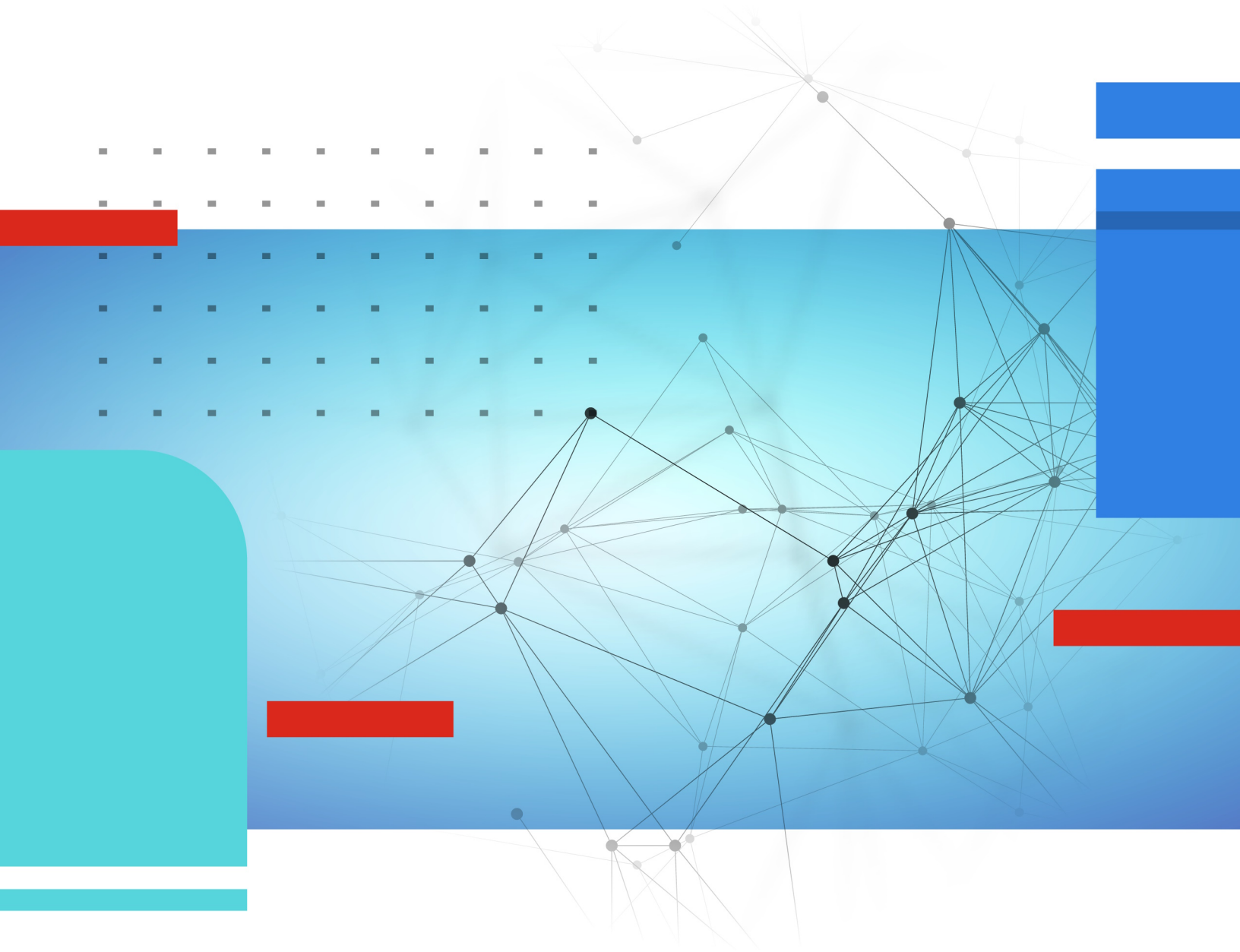
# HA Deployment Guide

**FortiClient EMS 7.4.4**



**FORTINET DOCUMENT LIBRARY**
https://docs.fortinet.com

**FORTINET VIDEO LIBRARY**
https://video.fortinet.com

**FORTINET BLOG**
https://blog.fortinet.com

**CUSTOMER SERVICE & SUPPORT**
https://support.fortinet.com

**FORTINET TRAINING & CERTIFICATION PROGRAM**
https://www.fortinet.com/training-certification

**FORTINET TRAINING INSTITUTE**
https://training.fortinet.com

**FORTIGUARD LABS**
https://www.fortiguard.com

**END USER LICENSE AGREEMENT**
https://www.fortinet.com/doc/legal/EULA.pdf

**FEEDBACK**
Email: techdoc@fortinet.com

January 15, 2026
FortiClient EMS 7.4.4 HA Deployment Guide
04-744-1175301-20260115

# TABLE OF CONTENTS

# Change log

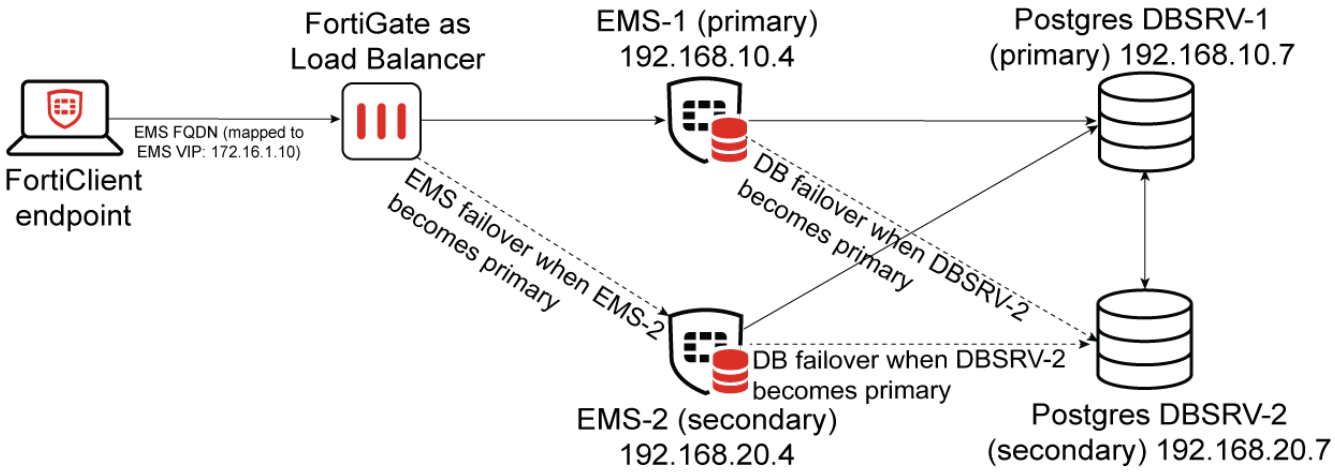
| Date | Change description |
|---|---|
| 2025-10-20 | Initial release. |
| 2025-10-22 | Updated the following:<br>• EMS nodes in HA with Postgres DB HA cluster for geo-redundancy on page 21<br>• EMS nodes in HA with Postgres DB HA cluster using EMS PostgreSQL HA Docker, single region on page 8<br>• EMS nodes in HA with Postgres DB HA cluster using native Postgres without Docker, single region on page 44 |
| 2025-11-03 | Updated EMS nodes in HA with Postgres DB HA cluster using native Postgres without Docker, single region on page 44. |
| 2025-11-10 | Updated EMS nodes in HA with Postgres DB HA cluster using native Postgres without Docker, single region on page 44. |
| 2025-11-18 | Updated EMS nodes in HA with Postgres DB HA cluster using EMS PostgreSQL HA Docker, single region on page 8. |
| 2025-12-15 | Updated EMS nodes in HA with Postgres DB HA cluster using EMS PostgreSQL HA Docker, single region on page 8. |
| 2026-01-12 | Updated EMS High Availability (HA) concepts on page 5. |
| 2026-01-15 | Updated EMS High Availability (HA) concepts on page 5. |

# EMS High Availability (HA) concepts

FortiClient EMS HA offers redundancy at both the application and database levels:

- EMS application HA—Two or more EMS nodes connect to the same database which can be a standalone PostgreSQL server or a list of PostgreSQL servers (DB nodes in short) that are part of a cluster.
- PostgreSQL DB HA—EMS connects to a list of PostgreSQL servers (DB nodes in short) that are part of a cluster.

## Topology




## EMS application HA

An EMS HA cluster can be formed when two or more EMS nodes connect to the same database, which can be a standalone PostgreSQL server or a list of PostgreSQL servers (DB nodes) that are part of a cluster. EMS supports only active-passive HA mode, where the first installed EMS node will be the primary and all subsequent EMS nodes will join the cluster as secondary or secondary nodes.

When deployed in HA mode, each EMS node registers itself in a database table where the node information gets recorded and in return gets all its configuration through the database which is shared across all nodes. There is no concept of "EMS node syncing" as all nodes get to look at the same data and configuration by using the same database and are therefore, always up to date.

The EMS primary node runs all EMS services and is the only node capable of receiving requests from endpoints. Whereas on the secondary nodes, only a few services run to perform a heartbeat for the node and to take action when the primary node becomes unavailable.

Each EMS node is assigned a random non-configurable priority value when joining the EMS cluster. When the primary node becomes unavailable, it stops performing its periodic heartbeat on the cluster registration database table. After a timeout, the secondary node with the highest priority that is currently online will be promoted to primary. The new primary node automatically starts up all services and will be able to receive traffic.

By default, each EMS node sends a heartbeat signal every 10 seconds. This interval can be configured using the *High Availability Keep Alive Interval* option (default is 10 seconds) in the *EMS Settings* page. When an EMS node does not send a heartbeat for more than twice the duration of the configured interval, that node is considered unavailable. If it is the primary up to that point, a new node will be promoted as the primary within up to another 60 seconds.

As only one EMS node is active at any given time while the remaining nodes are in standby mode, the sizing and capacity guidelines for managed endpoints (including CPU and RAM requirements per endpoint) in an EMS HA cluster are the same as for a single EMS instance, regardless of the number of EMS nodes in the HA cluster. See Management capacity in the *EMS Administration Guide* for details.

# PostgreSQL DB HA

EMS uses postgreSQL as a relational database, which is highly customizable and one of the most popular open-source databases. It is not a Fortinet product, nor supported by Fortinet.

You can configure EMS to connect to a standalone PostgreSQL server or a list of PostgreSQL servers (DB nodes) that are part of a cluster for better redundancy. When deployed in an HA cluster, PostgreSQL uses an addon, `repmgr` (Replication Manager) to replicate data from the primary node to the secondary nodes. The `repmgr` extension is also an open-source tool, and is not supported by Fortinet.

EMS requires a custom PostgreSQL extension to function. While instructions to install that extension are provided, we provide an EMS PostgreSQL HA `repmgr` custom docker image where that custom extension is already installed and ready for EMS. This image is further customized by Fortinet to include additional node reconciliation scripts to add another layer of protection and prevention of split brain. For this image, Fortinet provides support for matters related to replication, failover, split-brain and syncing between nodes but not for anything related to the database itself (PostgreSQL), docker, the host OS where the container runs on or network issues in the infrastructure that the cluster or DB nodes are part of.

When in HA, PostgreSQL only support Active-Passive mode where all secondary nodes are available in read-only mode and all connect to the primary node to replicate data from it using the `repmgr` addon. When the primary node becomes unavailable, the secondary nodes go into election to choose the new primary. The selection is based on the secondary node's individual priorities defined when setting up the cluster and needs to be agreed upon by a majority of secondary nodes. For a secondary node to be promoted to primary, it requires quorum of (n/2)+1, where n is the number of nodes in the cluster. This needs to be taken into account when setting up the PostgreSQL cluster to prevent a lock down where no other node can become a primary.

To prevent split brain scenarios where, due to communication loss, a node may think it needs to promote itself to primary while the primary is actually still active, additional special nodes (called witness) can be

added to help form a stronger quorum. While the witness nodes are part of the cluster, they do not take part in the data replication and are there only to monitor and vote on node promotion in case of failover.

The recommended number of nodes for a single region PostgreSQL HA cluster for a small cluster is 3 (2 DB nodes + 1 witness node). For a medium cluster, 5 (3 DB nodes + 2 witness nodes). For a large cluster, 9 (7 DB nodes + 2 witness nodes).

For EMS, the small cluster size is enough. The medium and large size should be used only depending on strong high availability requirements and infrastructure, but the larger the cluster, the more complex it is to setup and maintain. PostgreSQL with `repmgr` do provide Geo Redundancy support, but neither offer nor recommend automatic failover across regions.

When EMS connects to a PostgreSQL cluster, it connects and monitors a list of data nodes every 10 seconds to check which node is read-write (the primary). When there is a failover, EMS will check all the PostgreSQL DB nodes it knows of and will self update its configurations to point to the new primary.

The setup with the provided docker images is complex. They simplify the setup in most cases and sets up the recommended tools to make the cluster work, but are complex to maintain. It is recommended to deploy your own PostgreSQL cluster using a cloud service or setting up PostgreSQL nodes directly if you have of DBAs to maintain it. The table below compares these options in terms of support provided by Fortinet:

| | Cloud-managed PostgreSQL | Self-setup PostgreSQL | Fortinet-provided standalone Docker image | Fortinet-provided HA docker image |
|---|---|---|---|---|
| OS upgrades | No | No | No | No |
| PostgreSQL patching | No | No | Yes | Yes |
| Data backup / restore | No | No | No | No |
| Data storage | No | No | No | No |
| Docker daemon | - | - | No | No |
| EMS SQL errors related to data encryption | No | Yes | Yes | Yes |
| Server migration | - | No | No | No |
| Data replication | No | No | - | Yes* |

* Fortinet is only responsible for issues related to the monitor script shipped in the images and to ensure no customization-related issues are there. Most of the cases will be issues with `repmgr` or postgresql themselves, where Fortinet can point them out but not directly support.

See EMS HA deployment options on page 8 for a list of HA deployment options and detailed instructions for each scenario.

# EMS HA deployment options

FortiClient EMS supports the following HA setup options with EMS application HA combined with or without PostgreSQL DB HA (see EMS High Availability (HA) concepts on page 5):

| Install scenario | Description |
|---|---|
| EMS nodes in HA with Postgres DB HA cluster using EMS PostgreSQL HA Docker, single region on page 8 | Install EMS in an HA configuration with the Postgres DB also in an HA configuration using EMS PostgreSQL HA Docker. This install uses five Linux machines: two for the EMS HA nodes, two for the Postgres DB nodes, and one for the witness node. |
| EMS nodes in HA with Postgres DB HA cluster for geo-redundancy on page 21 | Install EMS nodes in a geographically distant HA cluster. Each data center also includes PostgreSQL DB/witness nodes that are part of the same cluster but running in different regions. The setup of a geographically distributed infrastructure allows you to ensure redundancy in case of failure on a data center. |
| Setting up HA for EMS VM appliances on page 36 | Set up an EMS cluster using two or more EMS VM appliances. Each EMS VM appliance, when deployed, is pre-installed with a local postgresql database which serves the EMS node in the appliance VM host. To set up an EMS VM cluster, all the EMS nodes within the cluster must be configured to use a remote postgresql database installed on none of the VM hosts. |
| EMS nodes in HA with standalone remote DB on page 39 | Install EMS in an HA configuration with the Postgres DB installed on a remote Linux machine. This install uses three Linux machines: two for the EMS HA nodes and one for the Postgres DB. |
| EMS nodes in HA with Postgres DB HA cluster using native Postgres without Docker, single region on page 44 | Install EMS in an HA configuration with the Postgres DB also in an HA configuration without using Docker. This install uses at least five servers: two for the EMS HA nodes and at least three for the Postgres DB/witness nodes. |

# EMS nodes in HA with Postgres DB HA cluster using EMS PostgreSQL HA Docker, single region

EMS requires a custom PostgreSQL extension to function and Fortinet provides an EMS PostgreSQL HA `repmgr` custom docker image where that custom extension is already installed and ready for EMS. This image is further customized by Fortinet to include additional node reconciliation scripts to add another layer of protection and prevention of split brain.

When using this image, Fortinet is responsible for providing support for matters related to replication, failover, split-brain, and syncing between nodes, but are not responsible nor supports anything related to the database itself (PostgreSQL), docker, the host OS where the container will be running or network issues in the infrastructure that the cluster or DB nodes are part of.

The following guide gives instructions on performing a fresh install of EMS HA with Postgres HA using EMS PostgreSQL HA Docker and for upgrading a previously installed EMS HA deployment with Postgres HA using EMS PostgreSQL HA Docker to 7.4.4:

- To configure EMS HA with Postgres HA using EMS PostgreSQL HA Docker: on page 9
- To upgrade a previously installed EMS deployment with Postgres HA using EMS PostgreSQL HA Docker to 7.4.4: on page 19

**To configure EMS HA with Postgres HA using EMS PostgreSQL HA Docker:**

1. Prepare five Linux machines. Three act as Postgres DB nodes (two for data nodes and one for the witness node) and the other two are EMS nodes.
   Refer to the EMS management capacity guide to plan the sizing of the data nodes.
2. If using cloned virtual machines for the EMS nodes, run the following commands on the cloned machine:

```
sudo rm /etc/machine-id
sudo rm /var/lib/dbus/machine-id
sudo systemd-machine-id-setup
```

3. Run `sudo -i` to log in to the shell with root privileges. Perform all following steps with root privileges.
4. Configure the Postgres HA cluster:
   a. Load the EMS Postgres HA Docker image in all Postgres DB nodes:
      i. Install Docker:

```
apt install docker.io
```

      ii. Download the Postgres Docker image `forticlientems_postgres-ha.tar.gz` file from the Fortinet Support site.
      iii. Load the image:

```
docker load -i forticlientems_postgres-ha.tar.gz
```

```
root@postgres1:/home/ems/Downloads# ls
ems_postgres-ha.tar.gz
root@postgres1:/home/ems/Downloads# sudo docker load -i ems_postgres-ha.tar.gz
7f19a690f50b: Loading layer [==================================================>]  280.9MB/280.9MB
259e04701d10: Loading layer [==================================================>]  19.97kB/19.97kB
34b3785e58cd: Loading layer [==================================================>]  19.46kB/19.46kB
74227982bbd0: Loading layer [==================================================>]  19.46kB/19.46kB
350f757aaff7: Loading layer [==================================================>]  19.46kB/19.46kB
30f9570f43bb: Loading layer [==================================================>]  22.02kB/22.02kB
1a1438813b9a: Loading layer [==================================================>]  4.608kB/4.608kB
a8ad2c286815: Loading layer [==================================================>]  32.26kB/32.26kB
63a09523f9fc: Loading layer [==================================================>]  4.608kB/4.608kB
9ebb0f072df7: Loading layer [==================================================>]  4.608kB/4.608kB
b7f7a4109629: Loading layer [==================================================>]   2.56kB/2.56kB
Loaded image: ems-postgres-ha:latest
root@postgres1:/home/ems/Downloads#
```

iv. List the images on Docker to verify the image has been created or loaded:

```
docker image ls
```

b. For each of the Postgres DB nodes (not the witness nodes), tune Postgres based on the host server specs by applying the recommended configuration as follows:

While there are various tools you can use to find the recommended configuration, the following instructions use PGTune to generate and copy the recommended configuration.

i. Create a folder called `conf.d` and configure the folder to be root owned with special permissions so it can be read by the containers:

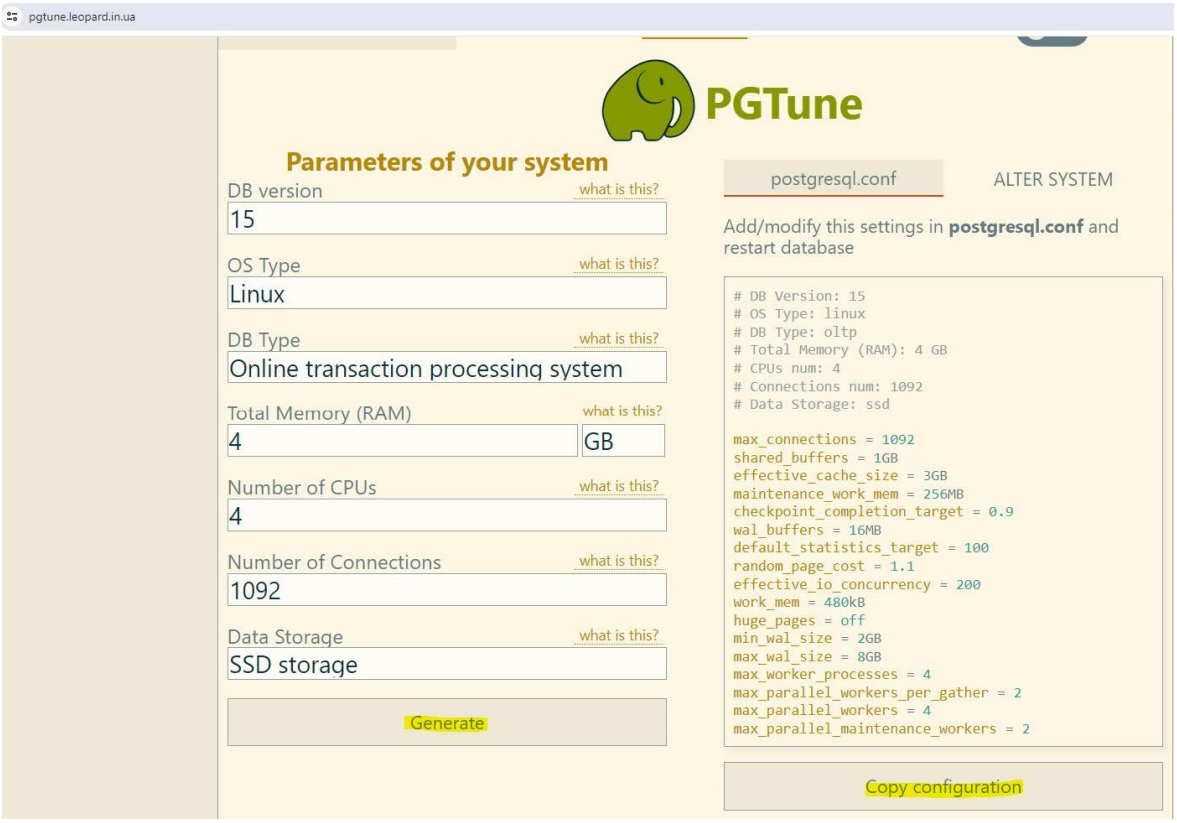
```
sudo chgrp -R root /path/to/conf.d/
sudo chmod -R g+rwX /path/to/conf.d/
```

ii. Create a file in the folder called `extra.conf`.

iii. Go to PGTune and enter the following information:

| Field | Value |
|---|---|
| *DB version* | 15 |
| *OS Type* | Linux |
| *DB Type* | Online transaction processing system |
| *Total Memory (RAM)* | Enter the total memory for your Postgres server. In this example, it is 4 GB. |
| *Number of CPUs* | Enter the total number of CPUs for your Postgres server. In this example, it is 4. |
| *Number of Connections* | 1092 |
| *Data Storage* | Enter the data storage type as per your device. In this example, it is SSD storage. |

iv. Click *Generate*.




v. Click *Copy configuration* and save the configuration to the `extra.conf` file you created earlier.

c. Start the primary DB:

i. Start the EMS PostgreSQL HA container on the primary DB node. In this example, the primary DB node is pg-1:

```
docker run --restart always --detach --name pg-1 -p 5432:5432 \
  --env POSTGRESQL_PASSWORD=<pg_password> \
  --env POSTGRESQL_DAEMON_USER=<pg_username> \
  --env REPMGR_PASSWORD=<pg_password> \
  --env REPMGR_PRIMARY_HOST=pg-1 \
  --env REPMGR_PRIMARY_PORT=5432 \
  --env REPMGR_PARTNER_NODES=pg-1,pg-2,pgw-1:5432 \
  --env REPMGR_NODE_NAME=pg-1 \
  --env REPMGR_NODE_NETWORK_NAME=pg-1 \
  --env REPMGR_PORT_NUMBER=5432 \
  --env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
  --env REPMGR_NODE_PRIORITY=100 \
  --volume pg_1_data:/bitnami/postgresql \
  --volume <path to conf.d>:/bitnami/postgresql/conf/conf.d \
  --shm-size 1g \
  --add-host pg-1:<ip/fqdn node 1> \
  --add-host pg-2:<ip/fqdn node 2> \
  --add-host pgw-1:<ip/fqdn witness 1 dc 1> \
ems-postgres-ha
```

- Replace <pg_password>, <pg_username>, and <path to conf.d> with your actual Postgres password, username, and path to conf.d file that you created earlier in the PostgreSQL configuration (such as /home/ems/conf.d).
- Replace <ip/fqdn xxxx> with the correct IP or FQDN to the corresponding PostgreSQL or witness node.

The following provides details for some of the parameters:

| Parameter | Value |
|---|---|
| REPMGR_PRIMARY_HOST | Hostname or IP address of where this node is running, which other nodes can access |
| REPMGR_PARTNER_NODES | Comma-separated list of the hostnames or IP addresses of all nodes that form this cluster |
| REPMGR_NODE_NAME | Node name. This can but does not necessarily need to match the hostname. This value shows when you perform step d to verify the cluster status. |
| REPMGR_NODE_NETWORK_NAME | Hostname or IP address of where this node is running. |

**ii.** Verify if the container is running:

```
docker ps -a
```

**iii.** Verify that the node has joined the cluster successfully:

```
docker exec -it pg-1 bash /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr
-f /opt/bitnami/repmgr/conf/repmgr.conf service status
```

**d.** From the secondary DB node, start the DB:

**i.** Start the EMS PostgreSQL HA container on the secondary node. In this example, the secondary node is pg-2:

```
docker run --restart always --detach --name pg-2 -p 5432:5432 \
  --env POSTGRESQL_PASSWORD=<pg_password> \
  --env POSTGRESQL_DAEMON_USER=<pg_username> \
  --env REPMGR_PASSWORD=<pg_password> \
  --env REPMGR_PRIMARY_HOST=pg-1 \
  --env REPMGR_PRIMARY_PORT=5432 \
  --env REPMGR_PARTNER_NODES=pg-1,pg-2,pgw-1:5432 \
  --env REPMGR_NODE_NAME=pg-2 \
  --env REPMGR_NODE_NETWORK_NAME=pg-2 \
  --env REPMGR_PORT_NUMBER=5432 \
  --env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
  --env REPMGR_NODE_PRIORITY=100 \
  --volume pg_2_data:/bitnami/postgresql \
  --volume <path to conf.d>:/bitnami/postgresql/conf/conf.d \
  --shm-size 1g \
  --add-host pg-1:<ip/fqdn node 1> \
  --add-host pg-2:<ip/fqdn node 2> \
```

```
  --add-host pgw-1:<ip/fqdn witness 1 dc 1> \
ems-postgres-ha
```

- Replace <pg_password>, <pg_username>, and <path to conf.d> with your actual Postgres password, username, and path to conf.d file that you created earlier in the PostgreSQL configuration (such as /home/ems/conf.d).
- Replace <ip/fqdn xxxx> with the correct IP or FQDN to the corresponding PostgreSQL or witness node.

The following provides details for some of the parameters:

| Parameter | Value |
|---|---|
| REPMGR_PRIMARY_HOST | Hostname or IP address of where the primary node is running, which other nodes can access |
| REPMGR_PARTNER_NODES | Comma-separated list of the hostnames or IP addresses of all nodes that form this cluster |
| REPMGR_NODE_NAME | Node name. This can but does not necessarily need to match the hostname. |
| REPMGR_NODE_NETWORK_NAME | Hostname or IP address of where this node is running |

**ii.** Verify if the container is running:

```
docker ps -a
```

**iii.** Verify that the node has joined the cluster successfully:

```
docker exec -it pg-2 bash /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr
-f /opt/bitnami/repmgr/conf/repmgr.conf service status
```

**e.** Start the PostgreSQL witness node:

The witness node does not take part in the data replication process and is a member of the cluster for the purposes of voting on node promotion. It requires as little as 2 GB RAM and 2 CPU.

**i.** Start the Postgres witness node. In this example, the witness node is pgw-1:

```
docker run --restart always --detach --name pgw-1 -p 5432:5432 \
  --env POSTGRESQL_PASSWORD=<pg_password> \
  --env POSTGRESQL_DAEMON_USER=<pg_username> \
  --env REPMGR_PASSWORD=<pg_password> \
  --env REPMGR_PRIMARY_HOST=pg-1 \
  --env REPMGR_PRIMARY_PORT=5432 \
  --env REPMGR_PARTNER_NODES=pg-1,pg-2,pgw-1:5432 \
  --env REPMGR_NODE_NAME=pgw-1 \
  --env REPMGR_NODE_NETWORK_NAME=pgw-1 \
  --env REPMGR_PORT_NUMBER=5432 \
```

```
  --env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
  --env REPMGR_NODE_TYPE=witness \
  --env REPMGR_NODE_ID_START_SEED=2000 \
  --volume pgw_1_data:/bitnami/postgresql \
  --shm-size 1g \
  --add-host pg-1:<ip/fqdn node 1> \
  --add-host pg-2:<ip/fqdn node 2> \
  --add-host pgw-1:<ip/fqdn witness 1 dc 1> \
ems-postgres-ha
```

- Replace <pg_password> and <pg_username> with your actual Postgres password and username.
- Replace <ip/fqdn xxxx> with the correct IP or FQDN to the corresponding PostgreSQL or witness node.

The following provides details for some of the parameters:

| Parameter | Value |
|---|---|
| REPMGR_PRIMARY_HOST | Hostname or IP address of where the primary node is running, which other nodes can access |
| REPMGR_PARTNER_NODES | Comma-separated list of the hostnames or IP addresses of all nodes that form this cluster |
| REPMGR_NODE_NAME | Node name. This can but does not necessarily need to match the hostname. |
| REPMGR_NODE_NETWORK_NAME | Hostname or IP address of where this node is running |

ii. Verify if the container is running:

```
docker ps -a
```

iii. Verify that the node has joined the cluster successfully:

```
docker exec -it pgw-1 bash /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr
-f /opt/bitnami/repmgr/conf/repmgr.conf service status
```

5. After the DB cluster is up and running, configure EMS HA:
   a. On both nodes, do the following:
      i. Download the forticlientems_7.4.4.2034.F.bin file from the Fortinet Support site.
      ii. Change permissions and add execute permissions to the installation file:
      chmod +x forticlientems_7.4.4.2034.F.bin
   b. On the primary node, install EMS:
      i. Set umask to 022 on file /etc/login.defs if the existing umask setting is more restrictive.
      ii. If you are installing EMS on Red Hat Enterprise Linux (RHEL) 9, do one of the following. EMS 7.4.4 and later versions support install on RHEL 9.:
      - If you are installing EMS on RHEL on Azure, run the following:

```
sudo dnf repolist enabled
```

Verify if `codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms` is in the list. If it is in the list, it is enabled. If it is not in the list, then run the following:

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms
```

Run the following commands:

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-$(rpm -E %rhel)-$(uname -m)/pgdg-redhat-repo-latest.noarch.rpm
sudo dnf config-manager --disable pgdg17 pgdg16
sudo dnf install -y https://rpms.remirepo.net/enterprise/remi-release-$(rpm -q --qf "%{VERSION}\n" redhat-release).rpm
sudo curl -o /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 https://rpms.remirepo.net/RPM-GPG-KEY-remi2021
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021
sudo ln -sf /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 /etc/pki/rpm-gpg/RPM-GPG-KEY-remi.el$(rpm -q --qf "%{VERSION}\n" redhat-release)
```

- If you are installing EMS on RHEL on AWS, run the following command:

```
sudo dnf config-manager --set-enabled codeready-builder-for-rhel-9-rhui-rpms
```

**iii.** Install EMS:

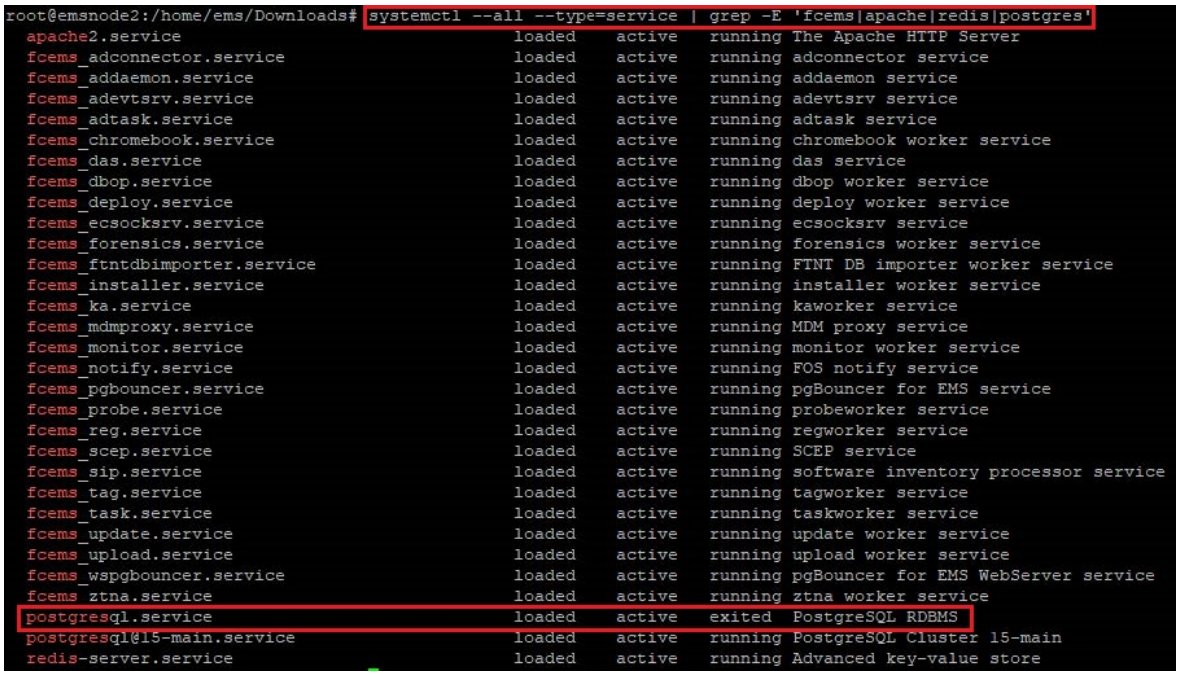
```
./forticlientems_7.4.4.2034.F.bin -- --db_host "<ip/fqdn node 1>,<ip/fqdn node 2>" --db_user postgres --db_pass postgres --skip_db_install --allowed_hosts '*' --enable_remote_https
```

Run the installer to and from any directory other than `/tmp`. Running the installer to or from `/tmp` causes issues.

Replace `<ip/fqdn xxxx>` with the correct IP address or FQDN of the corresponding DB node.

**iv.** After installation completes, check that all EMS services are running by entering the following command:

```
systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
```




The output shows that postgresql.service status displays as `exited`. This is the expected status. EMS does not create this service, which only exists to pass commands to version-specific Postgres services. It displays as part of the output as the command filters for all services that contain "postgres" in the name.

**c.** On the secondary node, install EMS:

**i.** Set `umask` to 022 if the existing `umask` setting is more restrictive.

**ii.** If you are installing EMS on Red Hat Enterprise Linux (RHEL) 9, do one of the following. EMS 7.4.4 and later versions support install on RHEL 9.:

- If you are installing EMS on RHEL on Azure, run the following:

```
sudo dnf repolist enabled
```

Verify if `codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms` is in the list. If it is in the list, it is enabled. If it is not in the list, then run the following:

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms
```

Run the following commands:

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-$(rpm -E %rhel)-$(uname -m)/pgdg-redhat-repo-latest.noarch.rpm
sudo dnf config-manager --disable pgdg17 pgdg16
sudo dnf install -y https://rpms.remirepo.net/enterprise/remi-release-$(rpm -q --qf "%{VERSION}\n" redhat-release).rpm
sudo curl -o /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 https://rpms.remirepo.net/RPM-GPG-KEY-remi2021
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021
sudo ln -sf /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 /etc/pki/rpm-gpg/RPM-GPG-KEY-remi.el$(rpm -q --qf "%{VERSION}\n" redhat-release)
```

- If you are installing EMS on RHEL on AWS, run the following command:

```
sudo dnf config-manager --set-enabled codeready-builder-for-rhel-9-rhui-rpms
```

iii. Install EMS:

```
./forticlientems_7.4.4.2034.F.bin -- --db_host "<ip/fqdn node 1>,<ip/fqdn node 2>" --
db_user postgres --db_pass postgres --skip_db_install --skip_db_deploy --allowed_hosts
'*' --enable_remote_https
```

Replace `<ip/fqdn xxxx>` with the correct IP address or FQDN of the corresponding DB node.

Run the installer to and from any directory other than `/tmp`. Running the installer to or from `/tmp` causes issues.

iv. After installation completes, check that EMS services are running by entering the following command. On the secondary EMS, only fcems_monitor, fcems_pgbouncer, fcems_wspgbouncer, and redis-server services should be running:

```
systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
```

d. After installation on both nodes completes, do the following:

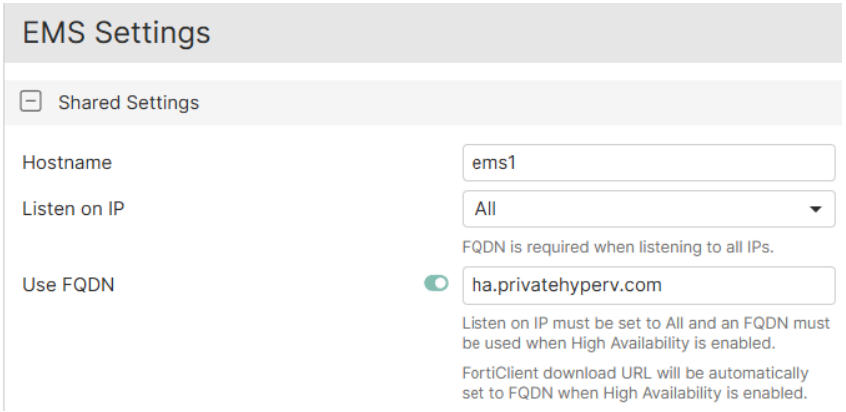
i. On the primary node, log in to EMS. License EMS as Starting FortiClient EMS and logging in describes. EMS HA requires a single license for the primary node and the secondary node(s). You only need to add the license to the primary node.

ii. Go to *System Settings > EMS Settings*.

iii. Under *Shared Settings*, confirm that *Listen on IP* is set to *All*.

iv. Enable *Use FQDN*.

v. Enter the desired FQDN.




vi. In the *Custom hostname* field, enter a virtual IP address (VIP) that is configured in the FortiGate load balancer (LB) as the VIP for EMS. In this example, the VIP is 172.16.1.50.

vii. Configure the *High Availability Keep Alive Internal* field with a value between 5 and 30 seconds.

viii. Go to *Dashboard > Status*. Confirm that the *System Information* widget displays that EMS is running in HA mode. If running in HA mode, the widget also lists the HA primary and secondary nodes and their statuses.

6. Configure a FortiGate as an LB for EMS HA:

a. Create a health check:
   i. Go to *Policy & Objects > Health Check*. Click *Create New*.
   ii. For *Type*, select *TCP*.
   iii. In the *Interval* field, enter 10.
   iv. In the *Timeout* field, enter 2.
   v. In the *Retry* field, enter 3.
   vi. In the *Port* field, enter 8013. Click *OK*.
b. Create a virtual server:
   i. Go to *Policy & Objects* and create a virtual server.
   ii. Configure the fields as follows:

| Field | Value |
|---|---|
| *Virtual server IP* | VIP that you configured in step 4.f. In this example, the VIP is 172.16.1.50. |
| *Virtual server port* | 10443 |
| *Load Balancing method* | *First Alive* |
| *Health check* | Monitor that you configured. |
| *Network Type* | *TCP* |

   iii. Under *Real Servers*, select *Create New*.
   iv. In the *IPv4 address* field, enter the primary EMS node IP address. In this example, it is 192.168.1.10.
   v. In the *Port* field, enter 10443.
   vi. In the *Max connections* field, enter 0.
   vii. For *Mode*, select *Active*.
   viii. Create a real server for the secondary EMS node. Click *Save*.
c. Repeat steps i-ix to create five additional virtual servers. The additional servers use ports 443, 8013, 8015, 8443, and 8871, but otherwise have identical settings to the first virtual server created. If you have enabled Chromebook management, create a virtual server for port 8443. Similarly, if you require importing an ACME certificate, create a virtual server for port 80.
d. Create a security policy that includes the LB virtual server as a destination address:
   i. Go to *Policy & Objects > Firewall Policy*.
   ii. Click *Create New*.
   iii. Configure the *Incoming Interface* and *Outgoing Interface* fields. The outgoing interface connects to the primary EMS node.
   iv. For *Source*, select *all*.
   v. In the *Destination* field, select ports 10443, 443, 8013, 8015, 8443, and 8871.
   vi. For *Service*, select *ALL*.
   vii. For *Inspection Mode*, select *Proxy-based*.
   viii. Save the policy.
   ix. If the EMS nodes are in different subnets, repeat these steps to configure a policy for the secondary EMS node. In this example, the nodes are in the same subnet, so you do not need to add a separate policy for the secondary EMS.

7. After the FortiGate LB configuration is complete, you can access EMS using the VIP configured in the FortiGate LB. If after initially installing EMS 7.4.4 you need to upgrade to a newer build, repeat steps 4.a.-c. with the new installation file.

**To upgrade a previously installed EMS deployment with Postgres HA using EMS PostgreSQL HA Docker to 7.4.4:**

If you are upgrading a previously installed EMS deployment with Postgres HA using EMS PostgreSQL HA (Bitnami) Docker from 7.4.3 or an earlier 7.4 version to 7.4.4, you must use the following procedure to ensure that your Postgres instances are upgraded without incurring data loss.

To upgrade your EMS nodes in a high availability (HA) cluster to 7.4.4, see Upgrading EMS in HA in the EMS Administration Guide.

1. Confirm that replication extensions are enabled on each node by running the following:

```
docker exec -ti pg-X psql -U postgres -c "ALTER SYSTEM SET shared_preload_libraries TO pg_
stat_statements, repmgr;"
```

   Replace pg-x with the actual Docker container name on each node. For example, the Docker container names may be pg-1, pg-2, and so on.
2. Stop the containers on each node:

```
docker stop pg-X
```

   Replace pg-x with the actual Docker container name on each node. For example, the Docker container names may be pg-1, pg-2, and so on.
3. Download the new image from the Fortinet Support portal and load it on the new primary node:

```
docker load -i forticlientems_7.4.4.2034.F_postgres-ha.tar.gz
```

4. Create a new instance (pgn-1 in this example) that points to the existing primary volume:

```
docker run --restart always --detach --name pgn-1 -p 5432:5432 \
  --env POSTGRESQL_PASSWORD=<pg_password> \
  --env POSTGRESQL_DAEMON_USER=<pg_username> \
  --env REPMGR_PASSWORD=<pg_password> \
  --env REPMGR_PRIMARY_HOST=pgn-1 \
  --env REPMGR_PRIMARY_PORT=5432 \
  --env REPMGR_PARTNER_NODES=pgn-1,pgn-2:5432 \
  --env REPMGR_NODE_NAME=pgn-1 \
  --env REPMGR_NODE_NETWORK_NAME=pgn-1 \
  --env REPMGR_PORT_NUMBER=5432 \
  --env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
  --env REPMGR_NODE_PRIORITY=100 \
  --env REPMGR_UPGRADE_EXTENSION=yes \
  --volume pg_1_data:/bitnami/postgresql \
  --volume <path to conf.d>:/bitnami/postgresql/conf/conf.d \
  --shm-size 1g \
  --add-host pgn-1:<ip/fqdn node 1> \
```

```
  --add-host pgn-2:<ip/fqdn node 2> \
ems-postgres-ha
```

- Replace <pg_password>, <pg_username>, and <path to conf.d> with your actual Postgres password, username, and path to conf.d file that you created earlier in the PostgreSQL configuration (such as /home/ems/conf.d).
- Replace <ip/fqdn xxxx> with the correct IP or FQDN to the corresponding PostgreSQL node.

5. Register the new primary node by updating the extensions to recognize the new node configurations:

```
docker exec -it pgn-1 /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr -f
/opt/bitnami/repmgr/conf/repmgr.conf primary register --force
```

6. Download the new image from the Fortinet Support portal and load it on the new secondary node:

```
docker load -i forticlientems_7.4.4.2034.F_postgres-ha.tar.gz
```

7. Create a new instance (pgn-2 in this example) that points to the existing secondary volume:

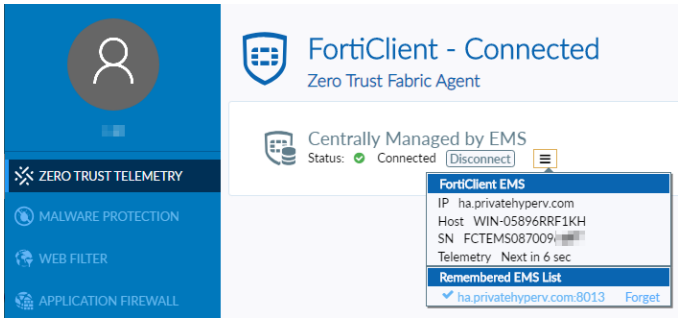
```
docker run --restart always --detach --name pgn-2 -p 5432:5432 \
  --env POSTGRESQL_PASSWORD=<pg_password> \
  --env POSTGRESQL_DAEMON_USER=<pg_username> \
  --env REPMGR_PASSWORD=<pg_password> \
  --env REPMGR_PRIMARY_HOST=pgn-1 \
  --env REPMGR_PRIMARY_PORT=5432 \
  --env REPMGR_PARTNER_NODES=pgn-1,pgn-2:5432 \
  --env REPMGR_NODE_NAME=pgn-2 \
  --env REPMGR_NODE_NETWORK_NAME=pgn-2 \
  --env REPMGR_PORT_NUMBER=5432 \
  --env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
  --env REPMGR_NODE_PRIORITY=100 \
  --env REPMGR_UPGRADE_EXTENSION=yes \
  --volume pg_1_data:/bitnami/postgresql \
  --volume <path to conf.d>:/bitnami/postgresql/conf/conf.d \
  --shm-size 1g \
  --add-host pgn-1:<ip/fqdn node 1> \
  --add-host pgn-2:<ip/fqdn node 2> \
ems-postgres-ha
```

- Replace <pg_password>, <pg_username>, and <path to conf.d> with your actual Postgres password, username, and path to conf.d file that you created earlier in the PostgreSQL configuration (such as /home/ems/conf.d).
- Replace <ip/fqdn xxxx> with the correct IP or FQDN to the corresponding PostgreSQL node.

**To validate the EMS HA configuration:**

1. Go to *Deployment & Installers* > *Manage Deployment*. Create a deployment package to deploy FortiClient to endpoints. See Adding a FortiClient installer.

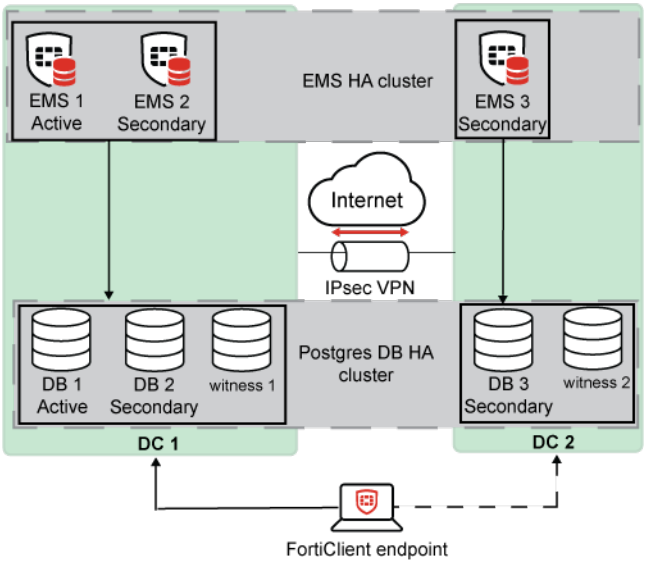
2. On an endpoint, download the deployment package from the download link.
3. Install FortiClient on the endpoint.
4. Ensure that FortiClient can register to the EMS server successfully using the FQDN.
5. Simulate HA by stopping *FortiClient Endpoint Management Server Monitor Service* on the primary EMS node. Ensure that the secondary EMS node is now the primary.
6. Ensure that FortiClient can still register to the EMS successfully using the FQDN.




# EMS nodes in HA with Postgres DB HA cluster for geo-redundancy

The geo-redundant deployment consists of EMS nodes in a geographically distant HA cluster and postgreSQL DB/witness nodes that are also part of the same cluster but running in different regions. The EMS nodes geo-redundancy provides HA in the application side: if all EMS nodes in one region become unavailable, one of the nodes from the other region becomes primary and still serves requests. From the database side, where data replication and other constraints make it more complex, the geo-redundant deployment focuses primarily on providing disaster recovery.

The following shows an example topology for this deployment:



- Two geographically distant data centers: DC1 (primary) and DC2 (secondary), with the following components:
  - An EMS HA cluster with three EMS nodes: two in DC1 and one in DC2. The configuration syncs between the nodes.
  - A Postgres DB HA cluster with five Postgres nodes: three in DC1 (two DB nodes and one witness node) and two in DC2 (one DB node and one witness node). Each DB node receives configuration data from the EMS node in its own data center without communicating with the DB nodes in the other data center.

    This setup uses the recommended number of nodes in the cluster to increase the likelihood that failover will be within the primary region and the secondary region is unlikely to be promoted automatically. Witness nodes are added to the Postgres DB cluster to prevent split brain scenarios where both nodes self-promote to become the primary and clash.

    When failover happens, witness nodes monitor and vote on node promotion without taking part in the data replication. If the failover causes the primary EMS node and primary DB node to be in different regions, performance will degrade and you may need to manually intervene to ensure that the primary EMS node and primary DB node are in the same region.

  - Each EMS and its Postgres DB are hosted on a VMware ESXi server.
- Site-to-site IPsec VPN connection
- 1 GB dedicated recommended site-to-site bandwidth
- FortiClient endpoints. The endpoint can access the EMS nodes in either data center. Typically, it accesses EMS 1. When failover occurs, it accesses EMS 2 or EMS 3.

**To configure EMS georedundant HA setup in the topology using the Fortinet-provided EMS PostgreSQL HA Docker:**

1. Verify that your configuration meets all EMS, Postgres, and Docker Swarm network requirements. See Required services and ports.
2. Prepare eight Linux machines. Five act as Postgres nodes (three for DB nodes and two for witness nodes) and the other three are EMS nodes.

   For the DB nodes, refer to the EMS management capacity guide to plan the sizing.

   The witness nodes require only 2 GB RAM and 2 CPUs as they do not take part in the data replication process and is a member of the cluster for the purpose of voting on node promotion in case of failover.

3. If using cloned virtual machines for the EMS nodes, run the following commands on the cloned machine:

   ```
   sudo rm /etc/machine-id
   sudo rm /var/lib/dbus/machine-id
   sudo systemd-machine-id-setup
   ```

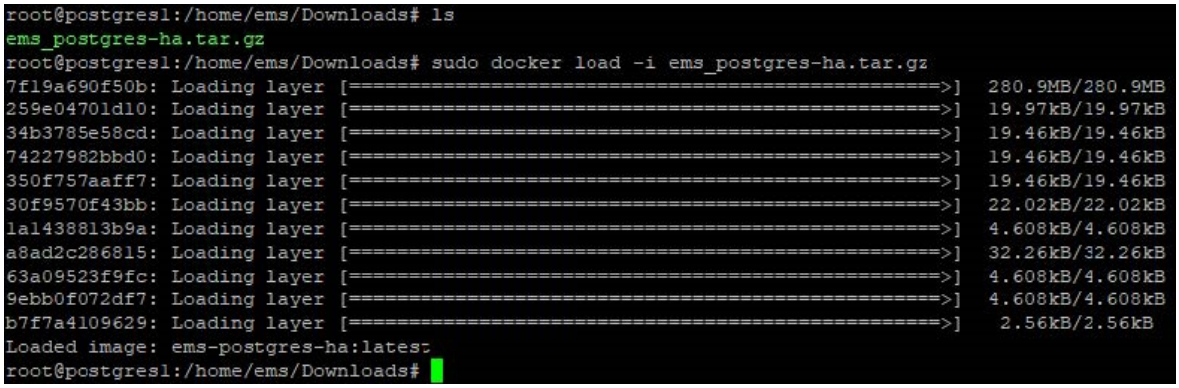
4. Run `sudo -i` to log in to the shell with root privileges. Perform all following steps with root privileges.

5. Configure the Postgres HA cluster in two regions:
   a. Load the EMS PostgreSQL HA Docker image in all Postgres nodes (including DB nodes and witness nodes):
      i. Install Docker:

```
apt install docker.io
```

      ii. Download the Postgres Docker image `forticlientems_postgres-ha.tar.gz` file from the Fortinet Support site.
      iii. Load the image:

```
docker load -i forticlientems_postgres-ha.tar.gz
```




      iv. List the images on Docker to verify the image has been created or loaded:

```
docker image ls
```

   b. For each of the DB nodes (not the witness nodes), tune Postgres based on the host server specs by applying the recommended configuration as follows:

While there are various tools you can use to find the recommended configuration, the following instructions use PGTune to generate and copy the recommended configuration.

      i. Create a folder called `conf.d` and configure the folder to be root owned with special permissions so it can be read by the containers:

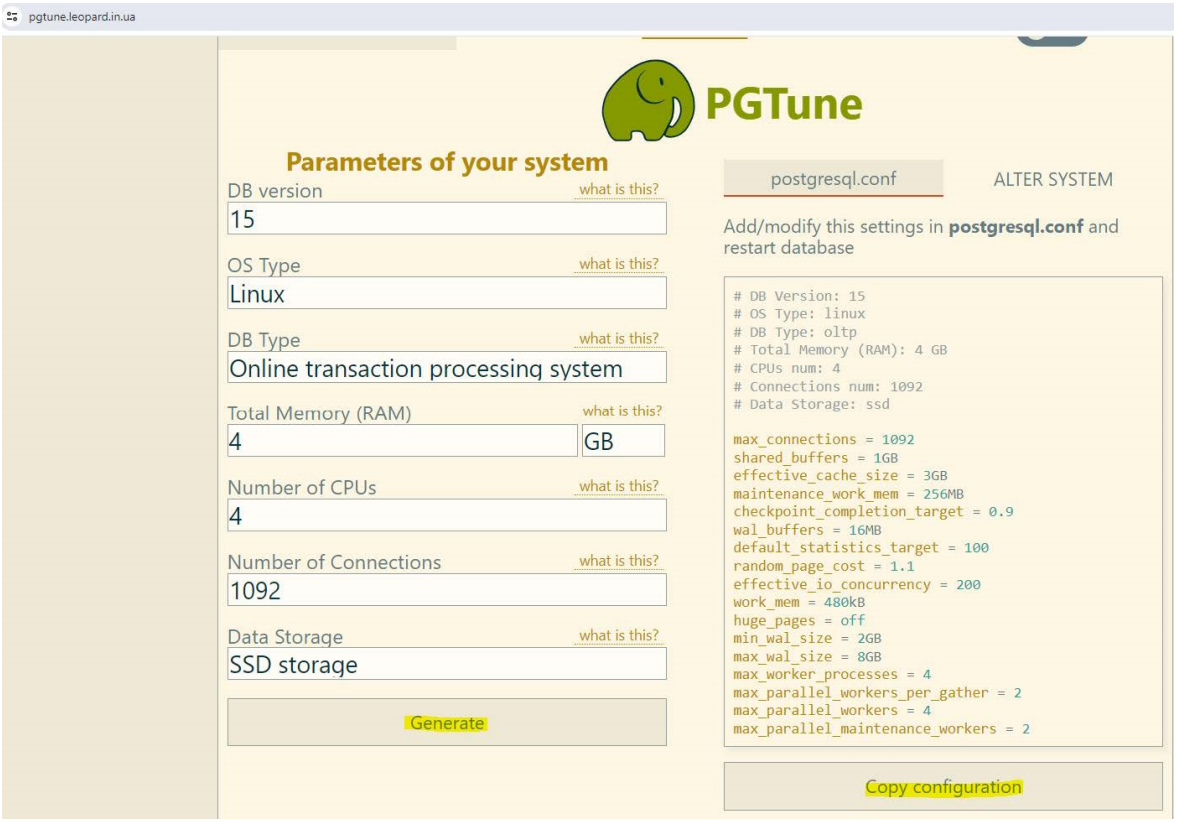
```
sudo chgrp -R root /path/to/conf.d/
sudo chmod -R g+rwX /path/to/conf.d/
```

      ii. Create a file in the folder called `extra.conf`.
      iii. Go to PGTune and enter the following information:

| Field | Value |
|---|---|
| *DB version* | 15 |
| *OS Type* | Linux |

| Field | Value |
|---|---|
| *DB Type* | Online transaction processing system |
| *Total Memory (RAM)* | Enter the total memory for your Postgres server. In this example, it is 4 GB. |
| *Number of CPUs* | Enter the total number of CPUs for your Postgres server. In this example, it is 4. |
| *Number of Connections* | 1092 |
| *Data Storage* | Enter the data storage type as per your device. In this example, it is SSD storage. |

iv. Click *Generate*.




v. Click *Copy configuration* and save the configuration to the `extra.conf` file you created earlier.

c. Start the primary DB in region 1:

i. Start the EMS PostgreSQL HA container on the primary node. In this example, the primary node is pg-1:

```
docker run --restart always --detach --name pg-1 -p 5432:5432 \
--env POSTGRESQL_PASSWORD=Fortinet123# \
--env POSTGRESQL_DAEMON_USER=postgres \
--env REPMGR_PASSWORD=Fortinet123# \
--env REPMGR_PRIMARY_HOST=pg-1 \
--env REPMGR_PRIMARY_PORT=5432 \
```

```
--env REPMGR_PARTNER_NODES=pg-1,pg-2,pgw-1,pg-3,pgw-2:5432 \
--env REPMGR_NODE_NAME=pg-1 \
--env REPMGR_NODE_NETWORK_NAME=pg-1 \
--env REPMGR_PORT_NUMBER=5432 \
--env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
--env REPMGR_NODE_PRIORITY=100 \
--env REPMGR_NODE_LOCATION=dc1 \
--volume pg_1_data:/bitnami/postgresql \
--volume /path/to/conf.d:/bitnami/postgresql/conf/conf.d \
--shm-size 1g \
--add-host pg-1:<ip/fqdn node 1> \
--add-host pg-2:<ip/fqdn node 2> \
--add-host pgw-1:<ip/fqdn witness 1 dc 1> \
--add-host pg-3:<ip/fqdn node 3> \
--add-host pgw-2:<ip/fqdn witness 2 dc 2> \
ems-postgres-ha
```

The example command configures Fortinet123# as the password. Ensure that you configure your own unique password as to not compromise the security of your installation.

Replace `/path/to/conf.d` with the correct path to the `conf.d` directory you created earlier in the PostgreSQL configuration.

Replace `<ip/fqdn xxxx>` with the correct IP or FQDN to the corresponding DB node or witness node.

The following provides details for some of the parameters:

| Parameter | Value |
|---|---|
| REPMGR_PRIMARY_HOST | Hostname or IP address of where this node is running, which other nodes can access |
| REPMGR_PARTNER_NODES | Comma-separated list of the hostnames or IP addresses of all nodes that form this cluster |
| REPMGR_NODE_NAME | Node name. This can but does not necessarily need to match the hostname. This value shows when you perform step d to verify the cluster status. |
| REPMGR_NODE_NETWORK_NAME | Hostname or IP address of where this node is running. |

ii. Verify if the container is running:

```
docker ps -a
```

iii. Verify that the node has joined the cluster successfully:

```
docker exec -it pg-1 bash /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr
-f /opt/bitnami/repmgr/conf/repmgr.conf service status
```

**d.** From the secondary DB node, start the DB in region 1:

**i.** Start the EMS PostgreSQL HA container on the secondary node. In this example, the secondary node is pg-2:

```
docker run --restart always --detach --name pg-2 -p 5432:5432 \
--env POSTGRESQL_PASSWORD=Fortinet123# \
--env POSTGRESQL_DAEMON_USER=postgres \
--env REPMGR_PASSWORD=Fortinet123# \
--env REPMGR_PRIMARY_HOST=pg-1 \
--env REPMGR_PRIMARY_PORT=5432 \
--env REPMGR_PARTNER_NODES=pg-1,pg-2,pgw-1,pg-3,pgw-2:5432 \
--env REPMGR_NODE_NAME=pg-2 \
--env REPMGR_NODE_NETWORK_NAME=pg-2 \
--env REPMGR_PORT_NUMBER=5432 \
--env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
--env REPMGR_NODE_PRIORITY=100 \
--env REPMGR_NODE_LOCATION=dc1 \
--volume pg_2_data:/bitnami/postgresql \
--volume /path/to/conf.d:/bitnami/postgresql/conf/conf.d \
--shm-size 1g \
--add-host pg-1:<ip/fqdn node 1> \
--add-host pg-2:<ip/fqdn node 2> \
--add-host pgw-1:<ip/fqdn witness 1 dc 1> \
--add-host pg-3:<ip/fqdn node 3> \
--add-host pgw-2:<ip/fqdn witness 2 dc 2> \
ems-postgres-ha
```

The example command configures Fortinet123# as the password. Ensure that you configure your own unique password as to not compromise the security of your installation.

Replace `/path/to/conf.d` with the correct path to the `conf.d` directory you created earlier in the PostgreSQL configuration.

Replace `<ip/fqdn xxxx>` with the correct IP or FQDN to the corresponding DB node or witness node.

The following provides details for some of the parameters:

| Parameter | Value |
|---|---|
| REPMGR_PRIMARY_HOST | Hostname or IP address of where the primary node is running, which other nodes can access |
| REPMGR_PARTNER_NODES | Comma-separated list of the hostnames or IP addresses of all nodes that form this cluster |
| REPMGR_NODE_NAME | Node name. This can but does not necessarily need to match the hostname. |
| REPMGR_NODE_NETWORK_NAME | Hostname or IP address of where this node is running |

ii. Verify if the container is running:

```
docker ps -a
```

iii. Verify that the node has joined the cluster successfully:

```
docker exec -it pg-2 bash /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr
-f /opt/bitnami/repmgr/conf/repmgr.conf service status
```

e. Start the witness node in region 1:

i. Start the witness node in region 1. In this example, the witness node is pgw-1:

```
docker run --restart always --detach --name pgw-1 -p 5432:5432 \
--env POSTGRESQL_PASSWORD=Fortinet123# \
--env POSTGRESQL_DAEMON_USER=postgres \
--env REPMGR_PASSWORD=Fortinet123# \
--env REPMGR_PRIMARY_HOST=pg-1 \
--env REPMGR_PRIMARY_PORT=5432 \
--env REPMGR_PARTNER_NODES=pg-1,pg-2,pgw-1,pg-3,pgw-2:5432 \
--env REPMGR_NODE_NAME=pgw-1 \
--env REPMGR_NODE_NETWORK_NAME=pgw-1 \
--env REPMGR_PORT_NUMBER=5432 \
--env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
--env REPMGR_NODE_TYPE=witness \
--env REPMGR_NODE_ID_START_SEED=2000 \
--volume pgw_1_data:/bitnami/postgresql \
--add-host pg-1:<ip/fqdn node 1> \
--add-host pg-2:<ip/fqdn node 2> \
--add-host pgw-1:<ip/fqdn witness 1 dc 1> \
--add-host pg-3:<ip/fqdn node 3> \
--add-host pgw-2:<ip/fqdn witness 2 dc 2> \
ems-postgres-ha
```

The example command configures Fortinet123# as the password. Ensure that you configure your own unique password as to not compromise the security of your installation.

Replace `/path/to/conf.d` with the correct path to the `conf.d` directory you created earlier in the PostgreSQL configuration.

Replace `<ip/fqdn xxxx>` with the correct IP or FQDN to the corresponding DB node or witness node.

The following provides details for some of the parameters:

| Parameter | Value |
|---|---|
| REPMGR_PRIMARY_HOST | Hostname or IP address of where the primary node is running, which other nodes can access |
| REPMGR_PARTNER_NODES | Comma-separated list of the hostnames or IP addresses of all nodes that form this cluster |

| Parameter | Value |
|---|---|
| REPMGR_NODE_NAME | Node name. This can but does not necessarily need to match the hostname. |
| REPMGR_NODE_NETWORK_NAME | Hostname or IP address of where this node is running |

ii. Verify if the container is running:

```
docker ps -a
```

iii. Verify that the node has joined the cluster successfully:

```
docker exec -it pgw-1 bash /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr
-f /opt/bitnami/repmgr/conf/repmgr.conf service status
```

f. Start the secondary DB node in region 2:

i. Start the secondary DB node in region 2. In this example, the secondary DB node is pg-3:

```
docker run --restart always --detach --name pg-3 -p 5432:5432 \
--env POSTGRESQL_PASSWORD=Fortinet123# \
--env POSTGRESQL_DAEMON_USER=postgres \
--env REPMGR_PASSWORD=Fortinet123# \
--env REPMGR_PRIMARY_HOST=pg-1 \
--env REPMGR_PRIMARY_PORT=5432 \
--env REPMGR_PARTNER_NODES=pg-1,pg-2,pgw-1,pg-3,pgw-2:5432 \
--env REPMGR_NODE_NAME=pg-3 \
--env REPMGR_NODE_NETWORK_NAME=pg-3 \
--env REPMGR_PORT_NUMBER=5432 \
--env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
--env REPMGR_NODE_PRIORITY=50 \
--env REPMGR_NODE_LOCATION=dc2 \
--volume pg_3_data:/bitnami/postgresql \
--volume /path/to/conf.d:/bitnami/postgresql/conf/conf.d \
--shm-size 1g \
--add-host pg-1:<ip/fqdn node 1> \
--add-host pg-2:<ip/fqdn node 2> \
--add-host pgw-1:<ip/fqdn witness 1 dc 1> \
--add-host pg-3:<ip/fqdn node 3> \
--add-host pgw-2:<ip/fqdn witness 2 dc 2> \
ems-postgres-ha
```

The example command configures Fortinet123# as the password. Ensure that you configure your own unique password as to not compromise the security of your installation.

Replace `/path/to/conf.d` with the correct path to the `conf.d` directory you created earlier in the PostgreSQL configuration.

Replace `<ip/fqdn xxxx>` with the correct IP or FQDN to the corresponding DB node or witness node.

The following provides details for some of the parameters:

| Parameter | Value |
|---|---|
| REPMGR_PRIMARY_HOST | Hostname or IP address of where the primary node is running, which other nodes can access |
| REPMGR_PARTNER_NODES | Comma-separated list of the hostnames or IP addresses of all nodes that form this cluster |
| REPMGR_NODE_NAME | Node name. This can but does not necessarily need to match the hostname. |
| REPMGR_NODE_NETWORK_NAME | Hostname or IP address of where this node is running |

ii. Verify if the container is running:

```
docker ps -a
```

iii. Verify that the node has joined the cluster successfully:

```
docker exec -it pg-3 bash /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr
-f /opt/bitnami/repmgr/conf/repmgr.conf service status
```

g. Start the witness node in region 2:

i. Start the witness node in region 2. In this example, the witness node is pgw-2:

```
docker run --restart always --detach --name pgw-1 -p 5432:5432 \
--env POSTGRESQL_PASSWORD=Fortinet123# \
--env POSTGRESQL_DAEMON_USER=postgres \
--env REPMGR_PASSWORD=Fortinet123# \
--env REPMGR_PRIMARY_HOST=pg-1 \
--env REPMGR_PRIMARY_PORT=5432 \
--env REPMGR_PARTNER_NODES=pg-1,pg-2,pgw-1,pg-3,pgw-2:5432 \
--env REPMGR_NODE_NAME=pgw-1 \
--env REPMGR_NODE_NETWORK_NAME=pgw-1 \
--env REPMGR_PORT_NUMBER=5432 \
--env REPMGR_DEGRADED_MONITORING_TIMEOUT=-1 \
--env REPMGR_NODE_TYPE=witness \
--env REPMGR_NODE_ID_START_SEED=2000 \
--volume pgw_1_data:/bitnami/postgresql \
--add-host pg-1:<ip/fqdn node 1> \
--add-host pg-2:<ip/fqdn node 2> \
--add-host pgw-1:<ip/fqdn witness 1 dc 1> \
--add-host pg-3:<ip/fqdn node 3> \
--add-host pgw-2:<ip/fqdn witness 2 dc 2> \
ems-postgres-ha
```

The example command configures Fortinet123# as the password. Ensure that you configure your own unique password as to not compromise the security of your installation.

Replace `/path/to/conf.d` with the correct path to the `conf.d` directory you created earlier in the PostgreSQL configuration.

Replace `<ip/fqdn xxxx>` with the correct IP or FQDN to the corresponding DB node or witness node.

The following provides details for some of the parameters:

| Parameter | Value |
|---|---|
| `REPMGR_PRIMARY_HOST` | Hostname or IP address of where the primary node is running, which other nodes can access |
| `REPMGR_PARTNER_NODES` | Comma-separated list of the hostnames or IP addresses of all nodes that form this cluster |
| `REPMGR_NODE_NAME` | Node name. This can but does not necessarily need to match the hostname. |
| `REPMGR_NODE_NETWORK_NAME` | Hostname or IP address of where this node is running |

ii. Verify if the container is running:

```
docker ps -a
```

iii. Verify that the node has joined the cluster successfully:

```
docker exec -it pgw-2 bash /opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr
-f /opt/bitnami/repmgr/conf/repmgr.conf service status
```

6. After the PostgreSQL DB HA cluster is up and running, configure EMS HA with two EMS nodes in DC 1 (primary region) and one in region 2 (secondary region):
   a. On all nodes, do the following:
      i. Download the `forticlientems_7.4.4.2034.F.bin` file from the Fortinet Support site.
      ii. Change permissions and add execute permissions to the installation file:

```
chmod +x forticlientems_7.4.4.2034.F.bin
```

   b. On the primary EMS node in region 1, install EMS:
      i. Set `umask` to 022 on file `/etc/login.defs` if the existing `umask` setting is more restrictive.
      ii. If you are installing EMS on Red Hat Enterprise Linux (RHEL) 9, do one of the following. EMS 7.4.4 and later versions support install on RHEL 9.:
      - If you are installing EMS on RHEL on Azure, run the following:

```
sudo dnf repolist enabled
```

Verify if `codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms` is in the list. If it is in the list, it is enabled. If it is not in the list, then run the following:

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-9-x86_64-eus-
rhui-rpms
```

Run the following commands:

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-
$(rpm -E %rhel)-$(uname -m)/pgdg-redhat-repo-latest.noarch.rpm
sudo dnf config-manager --disable pgdg17 pgdg16
sudo dnf install -y https://rpms.remirepo.net/enterprise/remi-release-$(rpm -q --
qf "%{VERSION}\n" redhat-release).rpm
sudo curl -o /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 https://rpms.remirepo.net/RPM-
GPG-KEY-remi2021
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021
sudo ln -sf /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 /etc/pki/rpm-gpg/RPM-GPG-KEY-
remi.el$(rpm -q --qf "%{VERSION}\n" redhat-release)
```

- If you are installing EMS on RHEL on AWS, run the following command:

```
sudo dnf config-manager --set-enabled codeready-builder-for-rhel-9-rhui-rpms
```

**iii.** Install EMS:

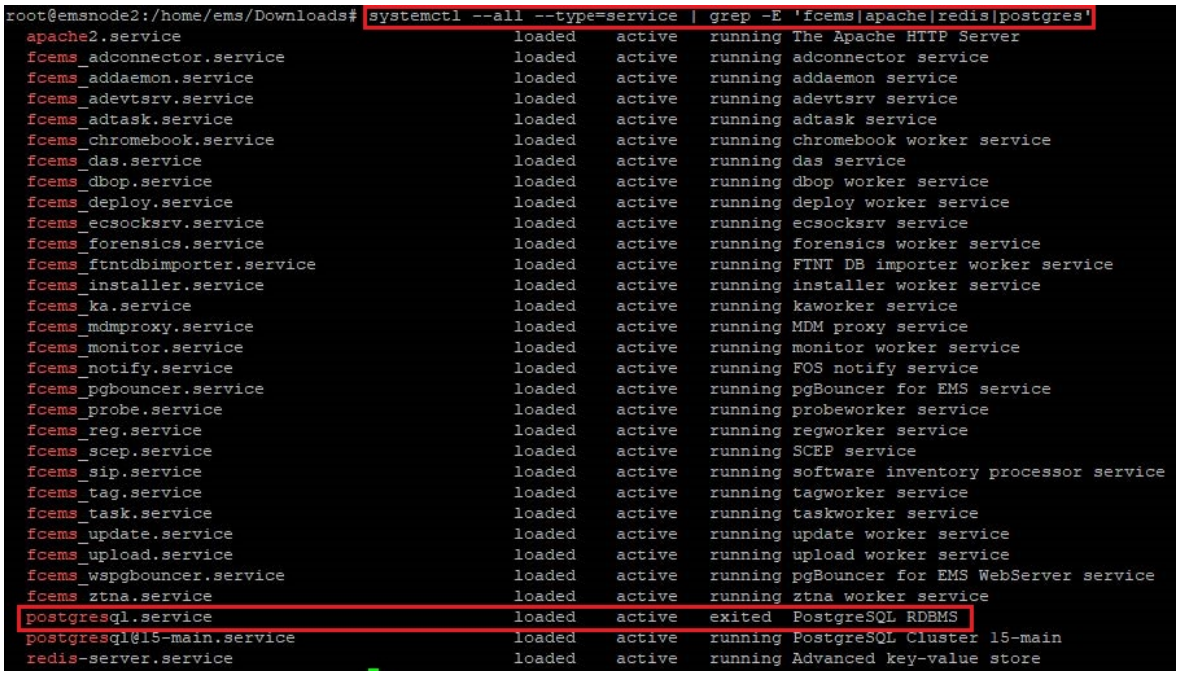
```
./forticlientems_7.4.4.2034.F.bin -- --db_host "<ip/fqdn node 1>,<ip/fqdn node
2>,<ip/fqdn node 3>" --db_user postgres --db_pass postgres --skip_db_install --
allowed_hosts '*' --enable_remote_https
```

Run the installer to and from any directory other than `/tmp`. Running the installer to or from `/tmp` causes issues.

Replace `<ip/fqdn xxxx>` with the correct IP address or FQDN of the corresponding DB node.

**iv.** After installation completes, check that all EMS services are running by entering the following command:

```
systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
```




The output shows that postgresql.service status displays as `exited`. This is the expected status. EMS does not create this service, which only exists to pass commands to version-specific Postgres services. It displays as part of the output as the command filters for all services that contain "postgres" in the name.

**c.** On each of the two secondary EMS nodes (one in each region), install EMS:

**i.** Set `umask` to 022 if the existing `umask` setting is more restrictive.

**ii.** If you are installing EMS on Red Hat Enterprise Linux (RHEL) 9, do one of the following. EMS 7.4.4 and later versions support install on RHEL 9.:

- If you are installing EMS on RHEL on Azure, run the following:

```
sudo dnf repolist enabled
```

Verify if `codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms` is in the list. If it is in the list, it is enabled. If it is not in the list, then run the following:

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms
```

Run the following commands:

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-$(rpm -E %rhel)-$(uname -m)/pgdg-redhat-repo-latest.noarch.rpm
sudo dnf config-manager --disable pgdg17 pgdg16
sudo dnf install -y https://rpms.remirepo.net/enterprise/remi-release-$(rpm -q --qf "%{VERSION}\n" redhat-release).rpm
sudo curl -o /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 https://rpms.remirepo.net/RPM-GPG-KEY-remi2021
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021
sudo ln -sf /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 /etc/pki/rpm-gpg/RPM-GPG-KEY-remi.el$(rpm -q --qf "%{VERSION}\n" redhat-release)
```

- If you are installing EMS on RHEL on AWS, run the following command:

```
sudo dnf config-manager --set-enabled codeready-builder-for-rhel-9-rhui-rpms
```

iii. Install EMS:

```
./forticlientems_7.4.4.2034.F.bin -- --db_host "<ip/fqdn node 1>,<ip/fqdn node
2>,<ip/fqdn node 3>" --db_user postgres --db_pass postgres --skip_db_install --skip_
db_deploy --allowed_hosts '*' --enable_remote_https
```

Replace `<ip/fqdn xxxx>` with the correct IP address or FQDN of the corresponding DB node.

Run the installer to and from any directory other than `/tmp`. Running the installer to or from `/tmp` causes issues.

iv. After installation completes, check that EMS services are running by entering the following command. On the secondary EMS, only fcems_monitor, fcems_pgbouncer, fcems_wspgbouncer, and redis-server services should be running:

```
systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
```

d. After installation on all nodes completes, do the following:

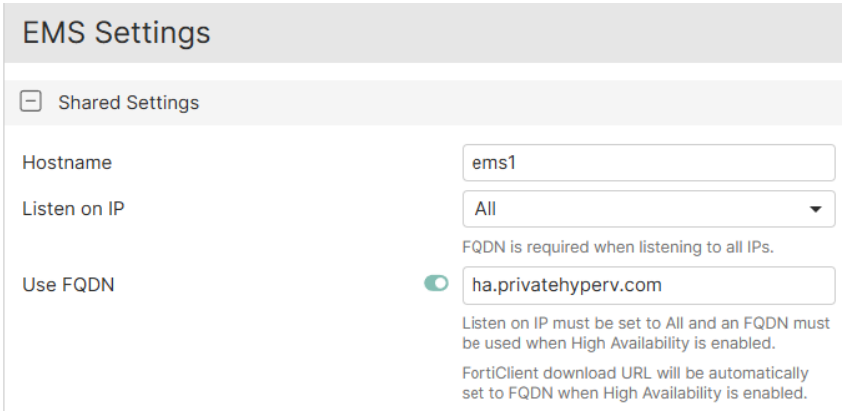
i. On the primary EMS node, log in to EMS. License EMS as Starting FortiClient EMS and logging in describes. EMS HA requires a single license for the primary node and the secondary node(s). You only need to add the license to the primary node.

ii. Go to *System Settings > EMS Settings*.

iii. Under *Shared Settings*, confirm that *Listen on IP* is set to *All*.

iv. Enable *Use FQDN*.

v. Enter the desired FQDN.




vi. In the *Custom hostname* field, enter a virtual IP address (VIP) that is configured in the FortiGate load balancer (LB) as the VIP for EMS. In this example, the VIP is 172.16.1.50.

vii. Configure the *High Availability Keep Alive Internal* field with a value between 5 and 30 seconds.

viii. Go to *Dashboard > Status*. Confirm that the *System Information* widget displays that EMS is running in HA mode. If EMS is running in HA mode, the widget also lists the EMS primary node and secondary nodes and their statuses.

7. Configure a FortiGate as an LB for EMS HA:

a. Create a health check:
   i. Go to *Policy & Objects > Health Check*. Click *Create New*.
   ii. For *Type*, select *TCP*.
   iii. In the *Interval* field, enter 10.
   iv. In the *Timeout* field, enter 2.
   v. In the *Retry* field, enter 3.
   vi. In the *Port* field, enter 8013. Click *OK*.
b. Create a virtual server:
   i. Go to *Policy & Objects* and create a virtual server.
   ii. Configure the fields as follows:

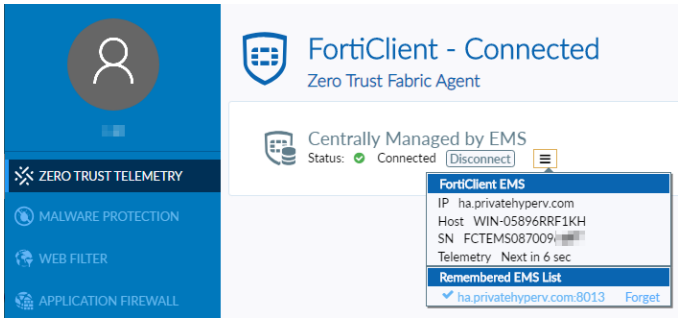
| Field | Value |
|---|---|
| *Virtual server IP* | VIP that you configured in step 4.f. In this example, the VIP is 172.16.1.50. |
| *Virtual server port* | 10443 |
| *Load Balancing method* | *First Alive* |
| *Health check* | Monitor that you configured. |
| *Network Type* | *TCP* |

   iii. Under *Real Servers*, select *Create New*.
   iv. In the *IPv4 address* field, enter the primary EMS node IP address. In this example, it is 192.168.1.10.
   v. In the *Port* field, enter 10443.
   vi. In the *Max connections* field, enter 0.
   vii. For *Mode*, select *Active*.
   viii. Create a real server for the secondary EMS node. Click *Save*.
c. Repeat steps i-ix to create five additional virtual servers. The additional servers use ports 443, 8013, 8015, 8443, and 8871, but otherwise have identical settings to the first virtual server created. If you have enabled Chromebook management, create a virtual server for port 8443. Similarly, if you require importing an ACME certificate, create a virtual server for port 80.
d. Create a security policy that includes the LB virtual server as a destination address:
   i. Go to *Policy & Objects > Firewall Policy*.
   ii. Click *Create New*.
   iii. Configure the *Incoming Interface* and *Outgoing Interface* fields. The outgoing interface connects to the primary EMS node.
   iv. For *Source*, select *all*.
   v. In the *Destination* field, select ports 10443, 443, 8013, 8015, 8443, and 8871.
   vi. For *Service*, select *ALL*.
   vii. For *Inspection Mode*, select *Proxy-based*.
   viii. Save the policy.
   ix. If the EMS nodes are in different subnets, repeat these steps to configure a policy for the secondary EMS node. In this example, the nodes are in the same subnet, so you do not need to add a separate policy for the secondary EMS.

8. After the FortiGate LB configuration is complete, you can access EMS using the VIP configured in the FortiGate LB. If after initially installing EMS 7.4.4 you need to upgrade to a newer build, repeat steps 4.a.-c. with the new installation file.

**To validate the EMS HA configuration:**

1. Go to *Deployment & Installers* > *Manage Deployment*. Create a deployment package to deploy FortiClient to endpoints. See Adding a FortiClient installer.
2. On an endpoint, download the deployment package from the download link.
3. Install FortiClient on the endpoint.
4. Ensure that FortiClient can register to the EMS server successfully using the FQDN.
5. Simulate HA by stopping *FortiClient Endpoint Management Server Monitor Service* on the primary EMS node. Ensure that the secondary EMS node is now the primary.
6. Ensure that FortiClient can still register to the EMS successfully using the FQDN.




Testing georedundancy and failover with a test endpoint is recommended before deployment to an EMS production environment.

### PostgreSQL DB HA failover scenarios

The proposed setup in this guide involves two regions with the following PostgreSQL nodes:

- **DC 1**: pg-1 (DB node), pg-2 (DB node) and pgw-1 (witness node)
- **DC 2**: pg-3 (DB node) and pgw-2 (witness node)

Assuming that nodes in DC 1 have priority 100 while nodes in DC 2 have priority 50. So regular node-based outages will favor failover between data nodes on DC 1. The table below elaborates the expected failover scenarios:

| Scenario | pg-1 (DC 1) status | pg-2 (DC 1) status | pg-3 (DC 2) status | Notes |
|---|---|---|---|---|
| pg-1 is the primary and becomes unavailable | unreachable | pg-2 becomes the primary | pg-3 keeps acting as secondary | There are enough nodes (pg-2, pgw-1, pg-3 and pgw2) for quorum. |

| Scenario | pg-1 (DC 1) status | pg-2 (DC 1) status | pg-3 (DC 2) status | Notes |
|---|---|---|---|---|
| pg-1 (the primary) and pg-2 become unavailable (pgw-1 still up) | unreachable | unreachable | pg-3 becomes the primary | There are enough nodes (pgw-1, pg-3 and pgw-2) for quorum. |
| pg-1 and pg-2 (the primary) become unavailable (pgw-1 still up) | unreachable | unreachable | pg-3 becomes the primary | There are enough nodes (pgw-1, pg-3 and pgw-2) for quorum. |
| DC 1 becomes unavailable (pg-1 was the primary) | unreachable | unreachable | pg-3 remains as secondary and consistently tries to reach out to pg-1 and pg2. **Needs manual promotion to primary.** | Not enough nodes for quorum (only pg-3 and pgw2 are online). |
| DC 2 becomes unavailable (pg-3 was the primary) | pg-1 becomes primary | pg-2 remains as secondary | unreachable | There are enough nodes (pgw-1, pg-1 and pg-2) for quorum. |
| DC 2 becomes unavailable (pg-3 was the primary) and pgw-1 is also unavailable | pg-1 remains as secondary. **Needs manual promotion to primary.** | pg-2 remains as secondary | unreachable | Not enough nodes for quorum (only pg-1 and pg-2 are online). |

**To manually promote a PostgreSQL DB node:**

```
docker exec -it <container name> bash /opt/bitnami/scripts/postgresql-repmgr/entry promote --force
```

Replace *<container name>* with the actual name of the container, e.g. pg-1, pg-2, etc.

# Setting up HA for EMS VM appliances

You can set up an EMS cluster using two or more EMS VM appliances. Each EMS VM appliance, when deployed, is pre-installed with a local postgresql database which serves the EMS node in the appliance VM host. To set up an EMS VM cluster, all the EMS nodes within the cluster must be configured to use a remote postgresql database installed on none of the VM hosts.

The following example configures two EMS VM nodes (EMS-1 and EMS-2) in an HA cluster. To add more EMS instances, repeat the procedure for each additional instance.

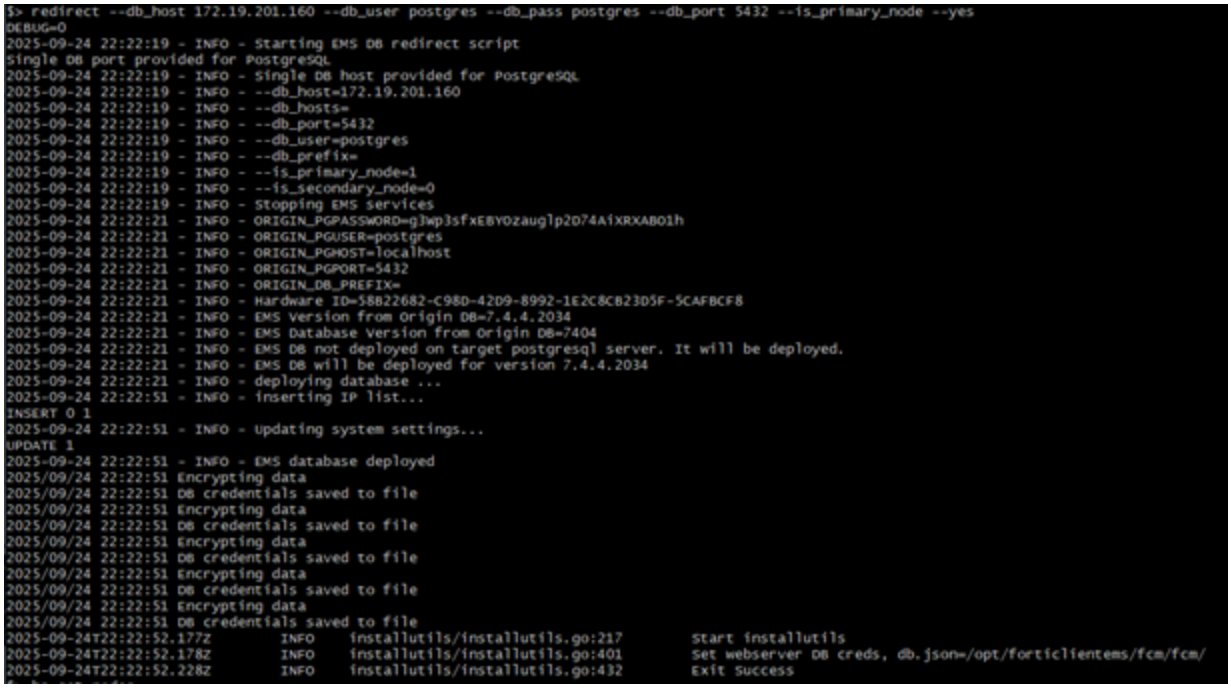
1. Prepare an EMS database on a postgresql remote server. See Installing EMS with standalone remote DB without Docker or EMS nodes in HA with standalone remote DB on page 39.
2. Deploy EMS-1 on a virtualization platform (e.g. Hyper-V, KVM, ESXi) and obtain the IP of EMS-1 host. See Deploying EMS as a VM image.
3. Verify that EMS-1 can reach the remote DB host using the following command:

   ```
   $> execute ping <IP of postgresql remote server host>
   ```

   If not, verify the network settings in your environment.
4. Deploy the EMS database on the remote postgresql database and make EMS-1 the primary node in the cluster using the `redirect` command:

   ```
   $> redirect --db_host <IP of postgresql remote server host> --db_pass <remote db password> --
   db_port <remote db port> --db_user <remote db username> --is_primary_node --yes
   ```

   **Sample output:**

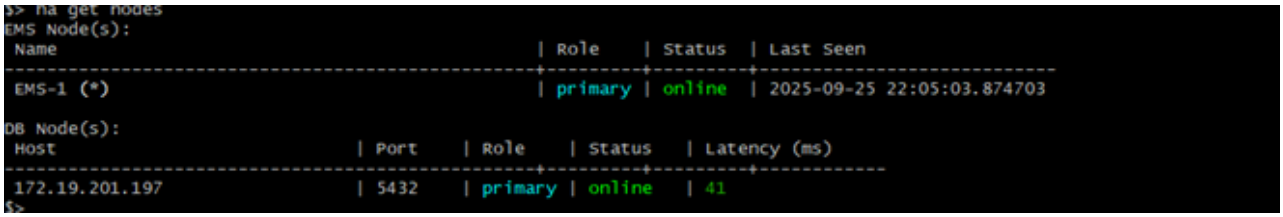
   
   
5. Access the EMS-1 console GUI via browser and complete initial configuration, such as username, password, and licenses. See Starting FortiClient EMS and logging in.
6. Update the hostname of EMS-1 using `system set hostname`:

   ```
   $> system set hostname "EMS-1"
   ```

7. Verify the cluster status using `ha get nodes`:

   ```
   $> ha get nodes
   ```

   **Sample output:**




It is expected that EMS-1 is shown as a standalone EMS node at this point because the second EMS node is not set yet.

8. Deploy EMS-2 on a virtualization platform (e.g. Hyper-V, KVM, ESXi) and obtain the IP of EMS-2 host. See Deploying EMS as a VM image.
9. Log in to the EMS-2 host by SSH using the user "ems" and the initial password "ems".

   You will be required to change the password later during the first login.
10. Verify that EMS-2 can reach the remote DB host using the following command:

```
$> execute ping <IP of postgresql remote server host>
```

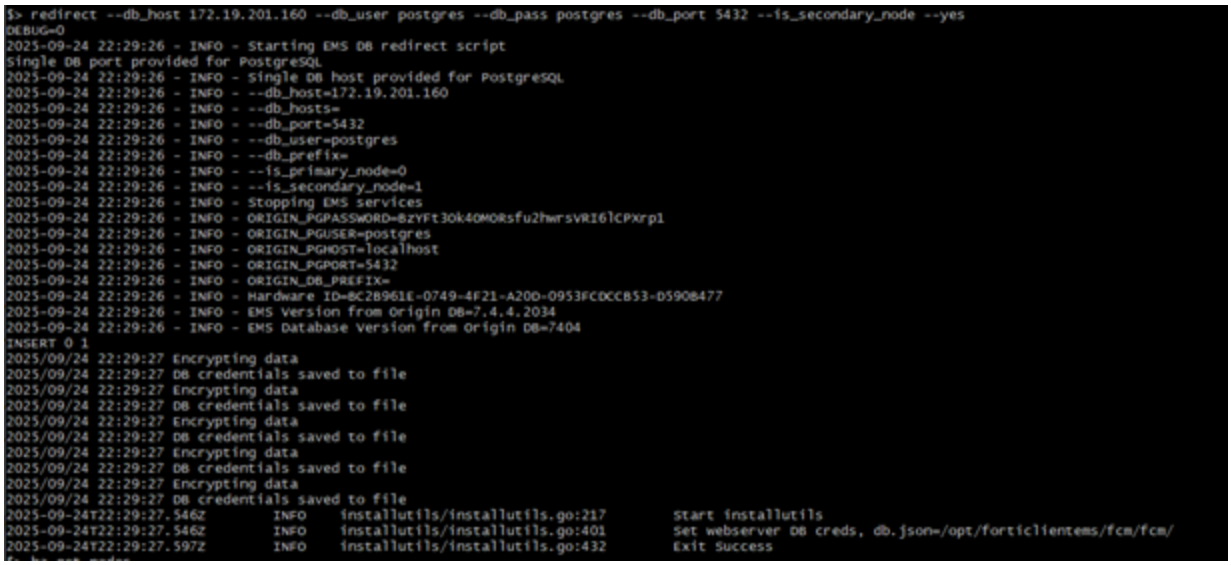
    If not, verify the network settings in your environment.
11. Update the hostname of EMS-2 using `system set hostname`:

```
$> system set hostname "EMS-2"
```

12. Execute a database redirection command on EMS-2 to allow access to the remote database and make EMS-2 the secondary node in the cluster using the `redirect` command:

```
$> redirect --db_host <IP of postgresql remote server host> --db_pass <remote db password> --
db_port <remote db port> --db_user <remote db username> --is_secondary_node --yes
```

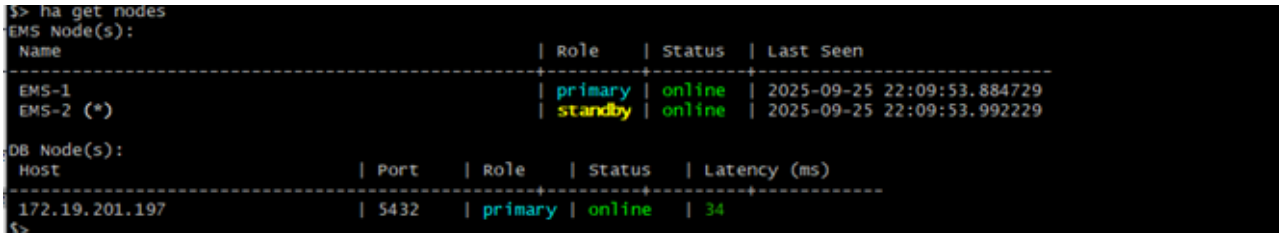
    **Sample output:**




13. Verify the configuration on EMS-2 and the cluster status using `ha get nodes`:

```
$> ha get nodes
```

    **Sample output:**




# EMS nodes in HA with standalone remote DB

The following procedure sets up an EMS HA cluster with the Postgres DB installed on a remote Linux machine. This setup does not use Docker and uses three Linux machines: two for the EMS nodes and one for the Postgres DB.

**To configure the Postgres DB, refer to the Postgres DB server configuration section in either of the following topics:**

- Installing EMS with Postgres in Docker
- Installing EMS with standalone remote DB without Docker

**To set up the EMS HA cluster:**

1. On both EMS nodes, do the following:
   a. Download the `forticlientems_7.4.4.2034.F.bin` file from https://support.fortinet.com.
   b. Change permissions and add execute permissions to the installation file:
   `sudo chmod +x forticlientems_7.4.4.2034.F.bin`
2. On the primary EMS node, install EMS:
   a. Set `umask` to `022` if the existing `umask` setting is more restrictive.
   b. If you are installing EMS on Red Hat Enterprise Linux (RHEL) 9, do one of the following. EMS 7.4.4 and later versions support install on RHEL 9.:
      - If you are installing EMS on RHEL on Azure, run the following:

```
sudo dnf repolist enabled
```

Verify if `codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms` is in the list. If it is in the list, it is enabled. If it is not in the list, then run the following:

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms
```

Run the following commands:

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-$(rpm -E %rhel)-$(uname -m)/pgdg-redhat-repo-latest.noarch.rpm
sudo dnf config-manager --disable pgdg17 pgdg16
sudo dnf install -y https://rpms.remirepo.net/enterprise/remi-release-$(rpm -q --qf "%
```

```
{VERSION}\n" redhat-release).rpm
sudo curl -o /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 https://rpms.remirepo.net/RPM-GPG-KEY-remi2021
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021
sudo ln -sf /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 /etc/pki/rpm-gpg/RPM-GPG-KEY-remi.el$(rpm -q --qf "%{VERSION}\n" redhat-release)
```

- If you are installing EMS on RHEL on AWS, run the following command:

```
sudo dnf config-manager --set-enabled codeready-builder-for-rhel-9-rhui-rpms
```

**c.** Install EMS:

**d.**

```
sudo ./forticlientems_7.4.4.2034.F.bin -- --db_host <IP address or FQDN> --db_port <local port> --db_user <username> --db_pass <password> --skip_db_install --allowed_hosts '*' --enable_remote_https
```

Run the installer to and from any directory other than `/tmp`. Running the installer to or from `/tmp` causes issues.

`db_host` is the remote Postgres DB server IP address.

**e.** After installation completes, check that all EMS services are running by entering the following command:

```
systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
```

```
root@emsnode2:/home/ems/Downloads# systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
  apache2.service                          loaded    active   running The Apache HTTP Server
  fcems_adconnector.service                loaded    active   running adconnector service
  fcems_addaemon.service                   loaded    active   running addaemon service
  fcems_adevtsrv.service                   loaded    active   running adevtsrv service
  fcems_adtask.service                     loaded    active   running adtask service
  fcems_chromebook.service                 loaded    active   running chromebook worker service
  fcems_das.service                        loaded    active   running das service
  fcems_dbop.service                       loaded    active   running dbop worker service
  fcems_deploy.service                     loaded    active   running deploy worker service
  fcems_ecsocksrv.service                  loaded    active   running ecsocksrv service
  fcems_forensics.service                  loaded    active   running forensics worker service
  fcems_ftntdbimporter.service             loaded    active   running FTNT DB importer worker service
  fcems_installer.service                  loaded    active   running installer worker service
  fcems_ka.service                         loaded    active   running kaworker service
  fcems_mdmproxy.service                   loaded    active   running MDM proxy service
  fcems_monitor.service                    loaded    active   running monitor worker service
  fcems_notify.service                     loaded    active   running FOS notify service
  fcems_pgbouncer.service                  loaded    active   running pgBouncer for EMS service
  fcems_probe.service                      loaded    active   running probeworker service
  fcems_reg.service                        loaded    active   running regworker service
  fcems_scep.service                       loaded    active   running SCEP service
  fcems_sip.service                        loaded    active   running software inventory processor service
  fcems_tag.service                        loaded    active   running tagworker service
  fcems_task.service                       loaded    active   running taskworker service
  fcems_update.service                     loaded    active   running update worker service
  fcems_upload.service                     loaded    active   running upload worker service
  fcems_wspgbouncer.service                loaded    active   running pgBouncer for EMS WebServer service
  fcems_ztna.service                       loaded    active   running ztna worker service
  postgresql.service                       loaded    active   exited  PostgreSQL RDBMS
  postgresql@15-main.service               loaded    active   running PostgreSQL Cluster 15-main
  redis-server.service                     loaded    active   running Advanced key-value store
```

The output shows that postgresql.service status displays as `exited`. This is the expected status. EMS does not create this service, which only exists to pass commands to version-specific Postgres services. It displays as part of the output as the command filters for all services that contain "postgres" in the name.

3. On the secondary EMS node, install EMS:
   a. Set `umask` to `022` if the existing `umask` setting is more restrictive.
   b. If you are installing EMS on Red Hat Enterprise Linux (RHEL) 9, do one of the following. EMS 7.4.4 and later versions support install on RHEL 9.:
      - If you are installing EMS on RHEL on Azure, run the following:

        ```
        sudo dnf repolist enabled
        ```

        Verify if `codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms` is in the list. If it is in the list, it is enabled. If it is not in the list, then run the following:

        ```
        sudo subscription-manager repos --enable codeready-builder-for-rhel-9-x86_64-eus-rhui-rpms
        ```

        Run the following commands:

        ```
        sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-$(rpm -E %rhel)-$(uname -m)/pgdg-redhat-repo-latest.noarch.rpm
        sudo dnf config-manager --disable pgdg17 pgdg16
        sudo dnf install -y https://rpms.remirepo.net/enterprise/remi-release-$(rpm -q --qf "%{VERSION}\n" redhat-release).rpm
        sudo curl -o /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 https://rpms.remirepo.net/RPM-GPG-KEY-remi2021
        sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021
        sudo ln -sf /etc/pki/rpm-gpg/RPM-GPG-KEY-remi2021 /etc/pki/rpm-gpg/RPM-GPG-KEY-remi.el$(rpm -q --qf "%{VERSION}\n" redhat-release)
        ```

      - If you are installing EMS on RHEL on AWS, run the following command:

        ```
        sudo dnf config-manager --set-enabled codeready-builder-for-rhel-9-rhui-rpms
        ```

   c. Install EMS:

      ```
      sudo ./forticlientems_7.4.4.2034.F.bin -- --db_host <IP address or FQDN> --db_port <local port> --db_user <username> --db_pass <password> --skip_db_install --skip_db_deploy --allowed_hosts '*' --enable_remote_https
      ```

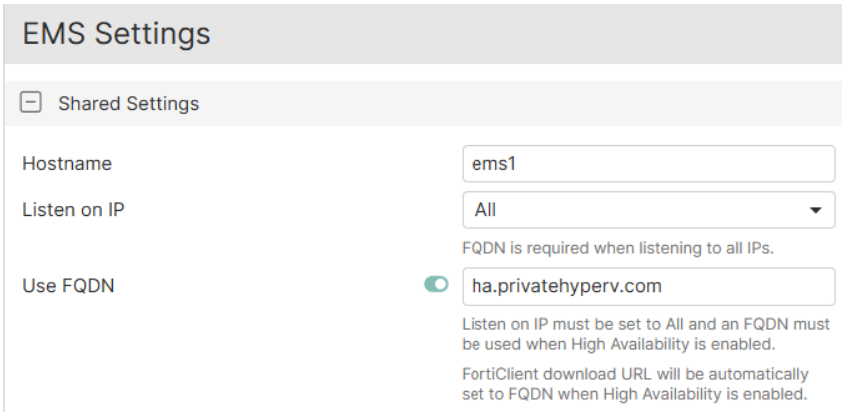
      Run the installer to and from any directory other than `/tmp`. Running the installer to or from `/tmp` causes issues.
   d. After installation completes, check that EMS services are running by entering the following command. On the secondary EMS node, only fcems_monitor, fcems_pgbouncer, fcems_wspgbouncer, and redis-server services should be running:

      ```
      systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
      ```

4. After installation on both nodes completes, do the following:
   a. On the primary node, log in to EMS. License EMS as Starting FortiClient EMS and logging in describes. EMS HA requires a single license for the primary node and the secondary node(s). You

only need to add the license to the primary node.

b. Go to *System Settings > EMS Settings*.
c. Under *Shared Settings*, confirm that *Listen on IP* is set to *All*.
d. Enable *Use FQDN*.
e. Enter the desired FQDN.




f. In the *Custom hostname* field, enter a virtual IP address (VIP) that is configured in the FortiGate load balancer (LB) as the VIP for EMS. In this example, the VIP is 172.16.1.50.
g. Configure the *High Availability Keep Alive Internal* field with a value between 5 and 30 seconds.
h. Go to *Dashboard > Status*. Confirm that the *System Information* widget displays that EMS is running in HA mode. If running in HA mode, the widget also lists the HA primary and secondary nodes and their statuses.

5. Go to *System Settings > EMS Settings*.
6. In the *Custom hostname* field, enter a virtual IP address (VIP) that is configured in the FortiGate load balancer (LB) as the VIP for EMS. In this example, the VIP is 172.16.1.50.
7. Configure a FortiGate as an LB for EMS HA:
   a. Create a health check:
      i. Go to *Policy & Objects > Health Check*. Click *Create New*.
      ii. For *Type*, select *TCP*.
      iii. In the *Interval* field, enter 10.
      iv. In the *Timeout* field, enter 2.
      v. In the *Retry* field, enter 3.
      vi. In the *Port* field, enter 8013. Click *OK*.
   b. Create a virtual server:
      i. Go to *Policy & Objects* and create a virtual server.
      ii. Configure the fields as follows:

| Field | Value |
|---|---|
| *Virtual server IP* | VIP that you configured in step 6. In this example, the VIP is 172.16.1.50. |
| *Virtual server port* | 10443 |
| *Load Balancing method* | *First Alive* |
| *Health check* | Monitor that you configured. |

iii. Under *Real Servers*, select *Create New*.
iv. In the *IPv4 address* field, enter the primary EMS node IP address. In this example, it is 192.168.1.10.
v. In the *Port* field, enter 10443.
vi. In the *Max connections* field, enter 0.
vii. For *Mode*, select *Active*.
viii. Create a real server for the secondary EMS node. Click *Save*.

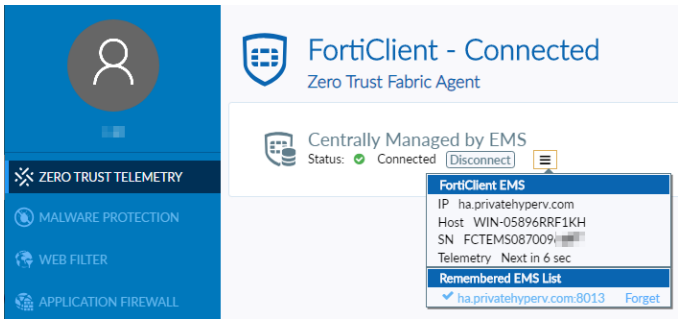
c. Repeat steps i-ix to create five additional virtual servers. The additional servers use ports 443, 8013, 8015, 8443, and 8871, but otherwise have identical settings to the first virtual server created. If you have enabled Chromebook management, create a virtual server for port 8443. Similarly, if you require importing an ACME certificate, create a virtual server for port 80.
d. Create a security policy that includes the LB virtual server as a destination address:
i. Go to *Policy & Objects > Firewall Policy*.
ii. Click *Create New*.
iii. Configure the *Incoming Interface* and *Outgoing Interface* fields. The outgoing interface connects to the primary EMS node.
iv. For *Source*, select *all*.
v. In the *Destination* field, select ports 10443, 443, 8013, 8015, 8443, and 8871.
vi. For *Service*, select *ALL*.
vii. For *Inspection Mode*, select *Proxy-based*.
viii. Save the policy.
ix. If the EMS nodes are in different subnets, repeat these steps to configure a policy for the secondary EMS node. In this example, the nodes are in the same subnet, so you do not need to add a separate policy for the secondary EMS.

8. After the FortiGate LB configuration is complete, you can access EMS using the VIP configured in the FortiGate LB. If after initially installing EMS 7.4.4 you need to upgrade to a newer build, repeat steps 1-3 with the new installation file.

**To validate the EMS HA configuration:**

1. Go to *Deployment & Installers > Manage Deployment*. Create a deployment package to deploy FortiClient to endpoints. See Adding a FortiClient installer.
2. On an endpoint, download the deployment package from the download link.
3. Install FortiClient on the endpoint.
4. Ensure that FortiClient can register to the EMS server successfully using the FQDN.
5. Simulate HA by stopping *FortiClient Endpoint Management Server Monitor Service* on the primary EMS node. Ensure that the secondary EMS node is now the primary.

6. Ensure that FortiClient can still register to the EMS successfully using the FQDN.




# EMS nodes in HA with Postgres DB HA cluster using native Postgres without Docker, single region

You can set up EMS nodes in HA with Postgres HA cluster where Postgres is natively installed (not Postgres containers).

This topic applies to Ubuntu Linux only. Issues may occur on other Linux distributions.

Fortinet generally recommends setting up DB nodes for HA implementation using the EMS PostgreSQL HA Docker image, as described in EMS nodes in HA with Postgres DB HA cluster using EMS PostgreSQL HA Docker, single region on page 8. See EMS High Availability (HA) concepts on page 5 for more information.

Postgres HA requires at least three servers or virtual machines (VM):

| Node | Purpose |
|---|---|
| DB node 1 | Starts as the primary node |
| DB node 2 | Starts as the secondary node |
| witness node | Serves as a witness to prevent split-brain scenarios where both nodes self-promote to become the primary and clash |

The configuration also requires two servers or VMs for EMS nodes.

**To configure DB node 1:**

1. Install Postgres 15 and Replication Manager. Postgres recommends this replication tool:

```
sudo apt install -y --no-install-recommends curl ca-certificates apt-transport-https

sudo install -d /usr/share/postgresql-common/pgdg
```

```
sudo curl -o /usr/share/postgresql-common/pgdg/apt.postgresql.org.asc --fail
https://www.postgresql.org/media/keys/ACCC4CF8.asc

sudo sh -c 'echo "deb [signed-by=/usr/share/postgresql-common/pgdg/apt.postgresql.org.asc]
https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list'

sudo apt update

sudo apt install -y postgresql-15 postgresql-15-repmgr
```

2. Install EMS custom extensions (`ems_pg_extensions.tar.gz`) for Postgres. You can download the extensions from the Fortinet Support site. Use the `ems_pg_extensions.tar.gz` file under the EMS 7.4.0 page to install extensions for EMS 7.4.4:

```
sudo tar zxvf ems_pg_extensions.tar.gz -C /
```

3. Create the symmetric key required that the custom extension requires. Copy the value in `symmetric_key.txt` as you must share it with the other nodes:

```
sh -c 'head -c 20 /dev/urandom | md5sum | head -c 20;' |sudo tee
/var/lib/postgresql/15/symmetric_key.txt > /dev/null
sudo chown postgres:postgres /var/lib/postgresql/15/symmetric_key.txt
cat /var/lib/postgresql/15/symmetric_key.txt
```

4. Set the password for the Postgres Linux user:

```
sudo passwd postgres
```

5. Modify `postgresql.conf` to add replication parameters. Change the following settings on `/etc/postgresql/15/main/postgresql.conf` with the indicated values:

```
listen_addresses = 'localhost,<PostgreSQL node1 server IP address>'
max_wal_senders = 10
max_replication_slots = 10
wal_level = replica
hot_standby = on
archive_mode = on
archive_command = '/bin/true'
shared_preload_libraries = 'repmgr'
```

6. Set up Postgres access to the replication user. Add the following to `/etc/postgresql/15/main/pg_hba.conf`:

```
host all all 127.0.0.1/32 trust
local replication repmgr trust
host replication repmgr 127.0.0.1/32 trust
host replication repmgr <DB node 1 ip>/32 trust
host replication repmgr <DB node 2 ip>/32 trust
host replication repmgr <witness node ip>/32 trust
local repmgr repmgr trust
```

```
host repmgr repmgr 127.0.0.1/32 trust
host repmgr repmgr <DB node 1 ip>/32 trust
host repmgr repmgr <DB node 2 ip>/32 trust
host repmgr repmgr <witness node ip>/32 trust
host all all 0.0.0.0/0 scram-sha-256
```

Replace `<DB node 1 ip>`, `<DB node 2 ip>`, and `<witness node ip>` accordingly.

7. Create the repmgr user and database. Run the following to create the repmgr user and database on Postgres:

```
sudo -u postgres createuser -s repmgr
sudo -u postgres createdb repmgr -O repmgr
```

8. Restart the postgresql service:

```
sudo systemctl restart postgresql@15-main.service
```

9. Set Postgres user password:

```
sudo -u postgres psql
ALTER USER postgres PASSWORD '<postgres password>';
```

**To configure DB node 2:**

1. Install Postgres 15 and Replication Manager. Postgres recommends this replication tool:

```
sudo apt install -y --no-install-recommends curl ca-certificates apt-transport-https

sudo install -d /usr/share/postgresql-common/pgdg

sudo curl -o /usr/share/postgresql-common/pgdg/apt.postgresql.org.asc --fail
https://www.postgresql.org/media/keys/ACCC4CF8.asc

sudo sh -c 'echo "deb [signed-by=/usr/share/postgresql-common/pgdg/apt.postgresql.org.asc]
https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list'

sudo apt update

sudo apt install -y postgresql-15 postgresql-15-repmgr
```

2. Install EMS custom extensions (`ems_pg_extensions.tar.gz`) for Postgres. You can download the extensions from the Fortinet Support site:

```
sudo tar zxvf ems_pg_extensions.tar.gz -C /
```

3. Create the symmetric key required that the custom extension requires. The value must be the one copied from the symmetric key from DB node 1:

```
sh -c 'echo "<value copied from DB node 1>"|head -c 20 | head -c 20;'|sudo tee
/var/lib/postgresql/15/symmetric_key.txt > /dev/null
```

4. Set the password for the Postgres Linux user:

```
sudo passwd postgres
```

5. Modify `postgresql.conf` to add replication parameters. Change the following settings on `/etc/postgresql/15/main/postgresql.conf` with the indicated values:

```
listen_addresses = 'localhost, <DB node 2 ip>'
max_wal_senders = 10
max_replication_slots = 10
wal_level = replica
hot_standby = on
archive_mode = on
archive_command = '/bin/true'
shared_preload_libraries = 'repmgr'
```

6. Set up Postgres access to the replication user. Add the following to `/etc/postgresql/15/main/pg_hba.conf`:

```
host all all 127.0.0.1/32 trust
local replication repmgr trust
host replication repmgr 127.0.0.1/32 trust
host replication repmgr <DB node 1 ip>/32 trust
host replication repmgr <DB node 2 ip>/32 trust
host replication repmgr <witness ip>/32 trust
local repmgr repmgr trust
host repmgr repmgr 127.0.0.1/32 trust
host repmgr repmgr <DB node 1 ip>/32 trust
host repmgr repmgr <DB node 2 ip>/32 trust
host repmgr repmgr <witness node ip>/32 trust
host all all 0.0.0.0/0 scram-sha-256
```

Replace `<DB node 1 ip>`, `<DB node 2 ip>`, and `<witness node ip>` accordingly.

7. Restart the postgresql service:

```
sudo systemctl restart postgresql@15-main.service
```

**To set up SSH access between DB node 1 and DB node 2:**

When the CLI prompts you, enter the postgres password.

1. On DB node 1, run the following:

```
sudo -H -u postgres bash -c "ssh-keygen -t rsa -f ~/.ssh/id_rsa -N ''"
sudo -H -u postgres bash -c "ssh-copy-id -i ~/.ssh/id_rsa.pub -o StrictHostKeyChecking=no
'postgres@<node2 ip>'"
```

2. On DB node 2, run the following:

```
sudo -H -u postgres bash -c "ssh-keygen -t rsa -f ~/.ssh/id_rsa -N ''"
sudo -H -u postgres bash -c "ssh-copy-id -i ~/.ssh/id_rsa.pub -o StrictHostKeyChecking=no
'postgres@<postgres node1 ip>'"
```

**To register DB node 1 to the cluster:**

1. Create the `/etc/repmgr.conf` file on DB node 1:

```
cluster= 'emscluster'
node_id=1
node_name=dbnode1
conninfo= 'host=<DB node 1 ip> user=repmgr dbname=repmgr connect_timeout=2'
data_directory= '/var/lib/postgresql/15/main/'
pg_bindir='/usr/bin/'
log_file= '/var/log/repmgr/repmgr.log'
log_level=DEBUG
failover=automatic
promote_command= '/usr/bin/repmgr standby promote -f /etc/repmgr.conf'
follow_command= '/usr/bin/repmgr standby follow -f /etc/repmgr.conf --upstream-node-id=%n'
```

Replace `<DB node 1 ip>` accordingly.

2. Create the log directory for repmgr:

```
sudo mkdir /var/log/repmgr
sudo chown -R postgres /var/log/repmgr
```

3. Register the node as the primary:

```
sudo -u postgres /usr/bin/repmgr -f /etc/repmgr.conf primary register
```

4. Confirm registration:

```
sudo -u postgres /usr/bin/repmgr -f /etc/repmgr.conf cluster show
```

5. Create a systemd service config for repmgr by creating the file `/etc/systemd/system/repmgrd.service`:

```
[Unit]
Description=PostgreSQL Replication Manager Daemon
After=network.target postgresql.service

[Service]
```

```
Type=forking
User=postgres
ExecStart=/usr/bin/repmgrd -f /etc/repmgr.conf --daemonize
PIDFile=/tmp/repmgrd.pid
ExecStop=/bin/kill -s TERM $(cat /tmp/repmgrd.pid)
ExecReload=/bin/kill -s HUP $(cat /tmp/repmgrd.pid)
Restart=on-failure
LimitNOFILE=16384

[Install]
WantedBy=multi-user.target
```

6. Edit `/etc/default/repmgrd` to change default settings. The following provides an example:

```
# default settings for repmgrd. This file is source by /bin/sh from
# /etc/init.d/repmgrd

# disable repmgrd by default so it won't get started upon installation
# valid values: yes/no
REPMGRD_ENABLED=yes

# configuration file (required)
REPMGRD_CONF="/etc/repmgr.conf"

# additional options
#REPMGRD_OPTS=“”
# user to run repmgrd as
#REPMGRD_USER=postgres

# repmgrd binary
#REPMGRD_BIN=/usr/bin/repmgrd

# pid file
REPMGRD_PIDFILE=/tmp/repmgrd.pid
```

`REPMGR_ENABLED` changed to `yes`, `REPMGRD_CONF` was uncommented and value changed to `"/etc/repmgrd.conf"` and `REPMGRD_PIDFILE` was uncommented and value changed to `/tmp/repmgrd.pid`.

7. Kill any previous instance of repmgrd that may be running:

```
ps -ef|grep “bin/repmgrd”|grep -v grep|xargs -t -i sudo kill {}
```

8. Enable and start the repmgrd service:

```
sudo systemctl enable repmgrd.service && sudo systemctl start repmgrd
```

9. Create a monitor and reconciliation service to support automatic failover. Create the script `/var/lib/postgresql/node_monitor.sh` with ownership to the Postgres user:

```
#!/bin/bash
```

```
# Script to detect a new primary and rejoin the cluster as secondary if necessary

# Configuration variables
REPMGR_CONF="/etc/repmgr.conf"
PG_SERVICE="postgresql@15-main"
CHECK_INTERVAL=10 # Check every 10 seconds
REPMGR_CMD="/usr/bin/repmgr"
PG_CTL="/usr/bin/pg_ctl"
NODE_NAME="node1"

# Function to log messages
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

# Main loop
while true; do
    # Ensure PostgreSQL is running before checking the cluster status
    systemctl is-active --quiet $PG_SERVICE
    PG_STATUS=$?

    if [ $PG_STATUS -eq 0 ]; then
        # Check if there is another primary in the cluster
        OTHER_PRIMARY=$(sudo -u postgres $REPMGR_CMD -f $REPMGR_CONF cluster show 2>/dev/null
| grep -v "$NODE_NAME" | grep "primary"|grep "running as primary"|sed -n 's/.*host=\([^
]*\).*/\1/p')

        # Get current node's role
        CURRENT_ROLE=$(sudo -u postgres $REPMGR_CMD -f $REPMGR_CONF node check --role
2>/dev/null | grep -oP '(?<=\().*(?=\))'|awk '{print $NF}')

        if [[ ! -z "$OTHER_PRIMARY" && "$CURRENT_ROLE" == "primary" ]]; then
            log_message "Another primary detected in the cluster. Attempting to rejoin as
secondary."

            # Ensure PostgreSQL is stopped before rejoining
            systemctl stop $PG_SERVICE
            sleep 5

            # Ensure PostgreSQL is fully stopped
            systemctl is-active --quiet $PG_SERVICE
            if [ $? -eq 0 ]; then
                log_message "Failed to stop PostgreSQL service. Skipping rejoin."
            else
                # Clone the data from the current primary and rejoin as secondary (standby)
                sudo -u postgres $REPMGR_CMD -h $OTHER_PRIMARY -U repmgr -d repmgr -f $REPMGR_
CONF standby clone --force 2>/dev/null && \
                sudo systemctl start $PG_SERVICE && \
                sudo -u postgres $REPMGR_CMD -f $REPMGR_CONF standby register --force
2>/dev/null

                if [ $? -eq 0 ]; then
```

```
                    log_message "Node rejoined as standby successfully."
                else
                    log_message "Failed to rejoin node as standby."
                fi
            fi
        elif [[ -z "$CLUSTER_STATUS" ]]; then
            log_message "No other primary detected, or the current node is already a standby.
Nothing to do."
        fi
    else
        log_message "PostgreSQL service is not running. Skipping checks."
    fi

    # Wait for the next check
    sleep $CHECK_INTERVAL
done
```

10. Add execute permissions to the `/var/lib/postgresql/node_monitor.sh` file:

```
sudo chmod +x node_monitor.sh
```

11. Create a monitor service by creating the file `/etc/systemd/system/pg_node_monitor.service`:

```
[Unit]
Description=Automatic Rejoin Standby Service
After=network.target postgresql.service

[Service]
Type=simple
User=root
ExecStart=/var/lib/postgresql/node_monitor.sh
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

12. Enable and start the monitor service:

```
sudo systemctl enable pg_node_monitor && sudo systemctl start pg_node_monitor
```

**To register DB node 2 to the cluster:**

1. Create the `/etc/repmgr.conf file` on DB node 2:

```
cluster='emscluster'
node_id=2
node_name=dbnode2
conninfo='host=<DB node 2 ip> user=repmgr dbname=repmgr connect_timeout=2'
data_directory='/var/lib/postgresql/15/main/'
pg_bindir='/usr/bin/'
```

```
log_file='/var/log/repmgr/repmgr.log'
log_level=DEBUG

failover=automatic
promote_command='/usr/bin/repmgr standby promote -f /etc/repmgr.conf'
follow_command='/usr/bin/repmgr standby follow -f /etc/repmgr.conf --upstream-node-id=%n'
```

Replace `<DB node 2 ip>` accordingly.

2. Create the log directory for repmgr:

```
sudo mkdir /var/log/repmgr
sudo chown -R postgres /var/log/repmgr
```

3. Stop the Postgres service:

```
sudo systemctl stop postgresql
```

4. Create a clone from DB node 1:

```
sudo -u postgres repmgr -h <DB node 1 ip> -U repmgr -d repmgr -f /etc/repmgr.conf standby
clone --force
```

Replace `<DB node 1 ip>` accordingly.

5. Restart the Postgres service:

```
sudo systemctl start postgresql
```

6. Register the node as the secondary (standby):

```
sudo -u postgres /usr/bin/repmgr -f /etc/repmgr.conf standby register
```

7. Confirm registration:

```
sudo -u postgres /usr/bin/repmgr -f /etc/repmgr.conf cluster show
```

8. Create a systemd service config for repmgr by creating file `/etc/systemd/system/repmgrd.service`:

```
[Unit]
Description=PostgreSQL Replication Manager Daemon
After=network.target postgresql.service

[Service]
Type=forking
User=postgres
ExecStart=/usr/bin/repmgrd -f /etc/repmgr.conf --daemonize
```

```
PIDFile=/tmp/repmgrd.pid
ExecStop=/bin/kill -s TERM $(cat /tmp/repmgrd.pid)
ExecReload=/bin/kill -s HUP $(cat /tmp/repmgrd.pid)
Restart=on-failure
LimitNOFILE=16384

[Install]
WantedBy=multi-user.target
```

9. Edit `/etc/default/repmgrd` to change default settings. The following provides an example:

```
# default settings for repmgrd. This file is source by /bin/sh from
# /etc/init.d/repmgrd

# disable repmgrd by default so it won't get started upon installation
# valid values: yes/no
REPMGRD_ENABLED=yes

# configuration file (required)
REPMGRD_CONF="/etc/repmgr.conf"

# additional options
#REPMGRD_OPTS=""

# user to run repmgrd as
#REPMGRD_USER=postgres

# repmgrd binary
#REPMGRD_BIN=/usr/bin/repmgrd

# pid file
REPMGRD_PIDFILE=/tmp/repmgrd.pid
```

`REPMGR_ENABLED` changed to `yes`, `REPMGRD_CONF` was uncommented and value changed to `"/etc/repmgrd.conf"` and `REPMGRD_PIDFILE` was uncommented and value changed to `/tmp/repmgrd.pid`.

10. Kill any previous instance of repmgrd that may be running:

```
ps -ef|grep “bin/repmgrd”|grep -v grep|xargs -t -i sudo kill {}
```

11. Enable and start the repmgrd service:

```
sudo systemctl enable repmgrd.service && sudo systemctl start repmgrd
```

12. Create a monitor and reconciliation service to support automatic failover. Create the script `/var/lib/postgresql/node_monitor.sh` with ownership to the Postgres user:

```
#!/bin/bash

# Script to detect a new primary and rejoin the cluster as secondary (standby) if necessary
```

```
# Configuration variables
REPMGR_CONF="/etc/repmgr.conf"
PG_SERVICE="postgresql@15-main"
CHECK_INTERVAL=10 # Check every 10 seconds
REPMGR_CMD="/usr/bin/repmgr"
PG_CTL="/usr/bin/pg_ctl"
NODE_NAME="node2"

# Function to log messages
log_message() {
       echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

# Main loop
while true; do
# Ensure PostgreSQL is running before checking the cluster status
systemctl is-active --quiet $PG_SERVICE
PG_STATUS=$?

if [ $PG_STATUS -eq 0 ]; then
       # Check if there is another primary in the cluster
       OTHER_PRIMARY=$(sudo -u postgres $REPMGR_CMD -f $REPMGR_CONF cluster show 2>/dev/null | grep
-v "$NODE_NAME" | grep "primary"|grep "running as primary"|sed -n 's/.*host=\([^ ]*\).*/\1/p')

       # Get current node's role
       CURRENT_ROLE=$(sudo -u postgres $REPMGR_CMD -f $REPMGR_CONF node check --role 2>/dev/null |
grep -oP '(?<=\().*(?=\))'|awk '{print $NF}')

       if [[ ! -z "$OTHER_PRIMARY" && "$CURRENT_ROLE" == "primary" ]]; then
               log_message "Another primary detected in the cluster. Attempting to rejoin as standby."

               # Ensure PostgreSQL is stopped before rejoining
               systemctl stop $PG_SERVICE
               sleep 5

               # Ensure PostgreSQL is fully stopped
               systemctl is-active --quiet $PG_SERVICE
               if [ $? -eq 0 ]; then
                       log_message "Failed to stop PostgreSQL service. Skipping rejoin."
               else
                       # Clone the data from the current primary and rejoin as secondary (standby)
                       sudo -u postgres $REPMGR_CMD -h $OTHER_PRIMARY -U repmgr -d repmgr -f $REPMGR_CONF st
clone --force 2>/dev/null && \
                       sudo systemctl start $PG_SERVICE && \
                       sudo -u postgres $REPMGR_CMD -f $REPMGR_CONF standby register --force 2>/dev/null

                       if [ $? -eq 0 ]; then
                               log_message "Node rejoined as standby successfully."
                       else
                               log_message "Failed to rejoin node as standby."
                       fi
               fi
```

```
       elif [[ -z "$CLUSTER_STATUS" ]]; then
               log_message "No other primary detected, or the current node is already a standby. Nothing to
do."
               fi
       else
               log_message "PostgreSQL service is not running. Skipping checks."
       fi
       # Wait for the next check
       sleep $CHECK_INTERVAL
done
```

13. Add execute permissions to the `/var/lib/postgresql/node_monitor.sh` file:

```
sudo chmod +x node_monitor.sh
```

14. Create a monitor service by creating the file `/etc/systemd/system/pg_node_monitor.service`:

```
[Unit]
Description=Automatic Rejoin Standby Service
After=network.target postgresql.service

[Service]
Type=simple
User=root
ExecStart=/var/lib/postgresql/node_monitor.sh
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

15. Enable and start the monitor service:

```
sudo systemctl enable pg_node_monitor && sudo systemctl start pg_node_monitor
```

**To register the witness node to the cluster:**

1. Install Postgres 15 and Replication Manager. Postgres recommends this replication tool:

```
sudo apt install -y --no-install-recommends curl ca-certificates apt-transport-https

sudo install -d /usr/share/postgresql-common/pgdg

sudo curl -o /usr/share/postgresql-common/pgdg/apt.postgresql.org.asc --fail
https://www.postgresql.org/media/keys/ACCC4CF8.asc

sudo sh -c 'echo "deb [signed-by=/usr/share/postgresql-common/pgdg/apt.postgresql.org.asc]
https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list'

sudo apt update
```

```
sudo apt install -y postgresql-15 postgresql-15-repmgr
```

2. Modify `postgresql.conf` to add replication parameters. Change the following settings on `/etc/postgresql/15/main/postgresql.conf` with the indicated values:

```
listen_addresses = 'localhost,<witness server node ip'
shared_preload_libraries = 'repmgr'
```

3. Set up Postgres access to the replication user. Add the following to `/etc/postgresql/15/main/pg_hba.conf`:

```
host all all 127.0.0.1/32 trust
local replication repmgr trust
host replication repmgr 127.0.0.1/32 trust
host replication repmgr <DB node 1 ip>/32 trust
host replication repmgr <DB node 2 ip>/32 trust
host replication repmgr <witness node ip>/32 trust
local repmgr repmgr trust
host repmgr repmgr 127.0.0.1/32 trust
host repmgr repmgr <DB node 1 ip>/32 trust
host repmgr repmgr <DB node 2 ip>/32 trust
host repmgr repmgr <witness node ip>/32 trust
host all all 0.0.0.0/0 scram-sha-256
```

Replace `<DB node 1 ip>`, `<DB node 2 ip>`, and `<witness node ip>` accordingly.

4. Restart the postgresql service:

```
sudo systemctl restart postgresql
```

5. Create the repmgr user and database. Run the following to create the repmgr user and database on Postgres:

```
sudo -u postgres createuser -s repmgr
sudo -u postgres createdb repmgr -O repmgr
```

6. Create the log directory for repmgr:

```
sudo mkdir /var/log/repmgr
sudo chown -R postgres /var/log/repmgr
```

7. Create the `/etc/repmgr.conf` file:

```
cluster='emscluster'
node_id=3
node_name='witness'
conninfo='host=<witness node ip> user=repmgr dbname=repmgr connect_timeout=2'
data_directory='/var/lib/postgresql/15/main' # Not used but required by repmgr
```

```
pg_bindir='/usr/bin/'
log_file='/var/log/repmgr/repmgr.log'
```

Replace `<witness node ip>` accordingly.

8. Register as witness informing the primary DB node, which is DB node 1:

```
sudo -u postgres /usr/bin/repmgr -f /etc/repmgr.conf witness register -h <DB node 1 ip>
```

Replace `<DB node 1 ip>` accordingly.

9. Create a systemd service config for repmgr by creating file `/etc/systemd/system/repmgrd.service`:

```
[Unit]
Description=PostgreSQL Replication Manager Daemon
After=network.target postgresql.service

[Service]
Type=forking
User=postgres
ExecStart=/usr/bin/repmgrd -f /etc/repmgr.conf --daemonize
PIDFile=/tmp/repmgrd.pid
ExecStop=/bin/kill -s TERM $(cat /tmp/repmgrd.pid)
ExecReload=/bin/kill -s HUP $(cat /tmp/repmgrd.pid)
Restart=on-failure LimitNOFILE=16384

[Install]
WantedBy=multi-user.target
```

10. Edit `/etc/default/repmgrd` to change default settings. It should look like this:

```
# default settings for repmgrd. This file is source by /bin/sh from
# /etc/init.d/repmgrd
# disable repmgrd by default so it won't get started upon installation
# valid values: yes/no
REPMGRD_ENABLED=yes

# configuration file (required)
REPMGRD_CONF="/etc/repmgr.conf"

# additional options
#REPMGRD_OPTS=""

# user to run repmgrd as
#REPMGRD_USER=postgres
```

```
# repmgrd binary
#REPMGRD_BIN=/usr/bin/repmgrd

# pid file
REPMGRD_PIDFILE=/tmp/repmgrd.pid
```

REPMGR_ENABLED changed to yes, REPMGRD_CONF was uncommented and value changed to "/etc/repmgrd.conf" and REPMGRD_PIDFILE was uncommented and value changed to /tmp/repmgrd.pid.

11. Kill any previous instance of repmgrd that might be running:

```
ps -ef|grep “bin/repmgrd”|grep -v grep|xargs -t -i sudo kill {}
```

12. Enable and start the repmgrd service:

```
sudo systemctl enable repmgrd.service && sudo systemctl start repmgrd
```

**To test DB failover:**

Stop the Postgres service on the primary DB node or shut down the primary DB node. By default, the health check happens every 10 seconds. Before promoting itself to the primary node, the secondary DB node checks if the primary is down at least six times. Therefore, failover takes 60 seconds. You can configure this value, but the default 60 seconds is an acceptable timeout or amount of downtime.

To check the failover status, run the following command on the secondary DB node:

```
sudo -u postgres /usr/bin/repmgr -f /etc/repmgr.conf cluster show
```

The secondary DB node (*Postgressqlnode2* in this example) is promoted as primary:

```
ID | Name            | Role    | Status    | Upstream        | Location | Priority | Timeline | Connection string
---+-----------------+---------+-----------+-----------------+----------+----------+----------+-------------------------------------------------------------
1  | postgresqlnode1 | primary | - failed  | ?               | default  | 100      |          | host=172.16.1.12 user=repmgr dbname=repmgr connect_timeout=2
2  | postgresqlnode2 | primary | * running |                 | default  | 100      | 2        | host=172.16.1.15 user=repmgr dbname=repmgr connect_timeout=2
3  | witness1        | witness | * running | postgresqlnode2 | default  | 0        | n/a      | host=172.16.1.22 user=repmgr dbname=repmgr connect_timeout=2
```

Start the postgresql service on the primary DB node (*Postgressqlnode1* in this example) or turn on the primary DB node. It is added as a secondary (standby). Run the following command to check the DB cluster status:

```
sudo -u postgres /usr/bin/repmgr -f /etc/repmgr.conf cluster show
```

```
ID | Name            | Role    | Status    | Upstream        | Location | Priority | Timeline | Connection string
---+-----------------+---------+-----------+-----------------+----------+----------+----------+-------------------------------------------------------------
1  | postgresqlnode1 | standby |   running | postgresqlnode2 | default  | 100      | 2        | host=172.16.1.12 user=repmgr dbname=repmgr connect_timeout=2
2  | postgresqlnode2 | primary | * running |                 | default  | 100      | 2        | host=172.16.1.15 user=repmgr dbname=repmgr connect_timeout=2
3  | witness1        | witness | * running | postgresqlnode2 | default  | 0        | n/a      | host=172.16.1.22 user=repmgr dbname=repmgr connect_timeout=2
```

**To configure EMS application HA:**

1. After the database cluster is up and running, configure EMS application HA:
   a. On both EMS nodes, do the following:
      i. Download the forticlientems_7.4.4.2034.F.bin file from the Fortinet Support site.
      ii. Run sudo -i to log in to the shell with root privileges.

iii. Change permissions and add execute permissions to the installation file:

```
chmod +x forticlientems_7.4.4.2034.F.bin
```

b. On the primary EMS node, install EMS:

i. Set `umask` to 022 on file `/etc/login.defs` if the existing `umask` setting is more restrictive.

ii. Install EMS:

```
./forticlientems_7.4.4.2034.F.bin -- --db_host "<DB node 1 ip>,<DB node 2 ip>" --db_
user postgres --db_pass <postgres password> --skip_db_install --allowed_hosts '*' --
enable_remote_https
```

Run the installer to and from any directory other than `/tmp`. Running the installer to or from `/tmp` causes issues.

Replace `<DB node 1 ip>` and `<DB node 2 ip>` with the correct IP address or FQDN of the corresponding DB node. Replace `<postgres password>` with the Postgres user password that you configured in the last step of To configure DB node 1: on page 44.

iii. After installation completes, check that all EMS services are running by entering the following command:

```
systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
```

```
root@emsnode2:/home/ems/Downloads# systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
  apache2.service                                loaded    active   running The Apache HTTP Server
  fcems_adconnector.service                      loaded    active   running adconnector service
  fcems_addaemon.service                         loaded    active   running addaemon service
  fcems_adevtsrv.service                         loaded    active   running adevtsrv service
  fcems_adtask.service                           loaded    active   running adtask service
  fcems_chromebook.service                       loaded    active   running chromebook worker service
  fcems_das.service                              loaded    active   running das service
  fcems_dbop.service                             loaded    active   running dbop worker service
  fcems_deploy.service                           loaded    active   running deploy worker service
  fcems_ecsocksrv.service                        loaded    active   running ecsocksrv service
  fcems_forensics.service                        loaded    active   running forensics worker service
  fcems_ftntdbimporter.service                   loaded    active   running FTNT DB importer worker service
  fcems_installer.service                        loaded    active   running installer worker service
  fcems_ka.service                               loaded    active   running kaworker service
  fcems_mdmproxy.service                         loaded    active   running MDM proxy service
  fcems_monitor.service                          loaded    active   running monitor worker service
  fcems_notify.service                           loaded    active   running FOS notify service
  fcems_pgbouncer.service                        loaded    active   running pgBouncer for EMS service
  fcems_probe.service                            loaded    active   running probeworker service
  fcems_reg.service                              loaded    active   running regworker service
  fcems_scep.service                             loaded    active   running SCEP service
  fcems_sip.service                              loaded    active   running software inventory processor service
  fcems_tag.service                              loaded    active   running tagworker service
  fcems_task.service                             loaded    active   running taskworker service
  fcems_update.service                           loaded    active   running update worker service
  fcems_upload.service                           loaded    active   running upload worker service
  fcems_wspgbouncer.service                      loaded    active   running pgBouncer for EMS WebServer service
  fcems_ztna.service                             loaded    active   running ztna worker service
  postgresql.service                             loaded    active   exited  PostgreSQL RDBMS
  postgresql@15-main.service                     loaded    active   running PostgreSQL Cluster 15-main
  redis-server.service                           loaded    active   running Advanced key-value store
```

The output shows that postgresql.service status displays as `exited`. This is the expected status. EMS does not create this service, which only exists to pass commands to version-specific Postgres services. It displays as part of the output as the command filters for all services that contain "postgres" in the name.

c. On the secondary EMS node, install EMS:
   i. Set `umask` to 022 if the existing `umask` setting is more restrictive.
   ii. Install EMS:

```
./forticlientems_7.4.4.2034.F.bin -- --db_host "<DB node 1 ip>,<DB node 2 ip>" --db_
user postgres --db_pass <postgres password> --skip_db_install --skip_db_deploy --
allowed_hosts '*' --enable_remote_https
```

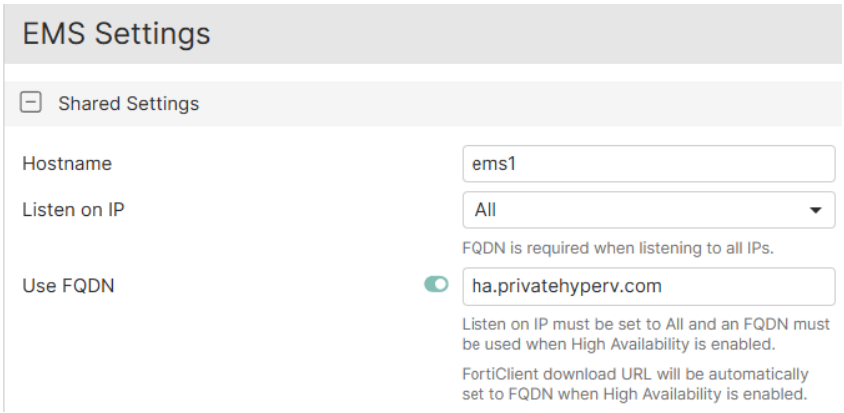
   Run the installer to and from any directory other than `/tmp`. Running the installer to or from `/tmp` causes issues.

   Replace `<DB node 1 ip>` and `<DB node 2 ip>` with the correct IP address or FQDN of the corresponding DB node. Replace `<postgres password>` with the Postgres user password that you configured in the last step of To configure DB node 1: on page 44.

   iii. After installation completes, check that EMS services are running by entering the following command. On the secondary EMS node, only `fcems_monitor`, `fcems_pgbouncer`, `fcems_wspgbouncer`, and `redis-server` services should be running:

```
systemctl --all --type=service | grep -E 'fcems|apache|redis|postgres'
```

d. After installation on both nodes completes, do the following:
   i. On the primary node, log in to EMS. License EMS as Starting FortiClient EMS and logging in describes. EMS HA requires a single license for the primary node and the secondary node(s). You only need to add the license to the primary node.
   ii. Go to *System Settings > EMS Settings*.
   iii. Under *Shared Settings*, confirm that *Listen on IP* is set to *All*.
   iv. Enable *Use FQDN*.
   v. Enter the desired FQDN.




   vi. In the *Custom hostname* field, enter a virtual IP address (VIP) that is configured in the FortiGate load balancer (LB) as the VIP for EMS. In this example, the VIP is 172.16.1.50.
   vii. Configure the *High Availability Keep Alive Internal* field with a value between 5 and 30 seconds.
   viii. Go to *Dashboard > Status*. Confirm that the *System Information* widget displays that EMS is running in HA mode. If running in HA mode, the widget also lists the HA primary and secondary nodes and their statuses.

2. Configure a FortiGate as an LB for EMS HA:

a. Create a health check:
   i. Go to *Policy & Objects > Health Check*. Click *Create New*.
   ii. For *Type*, select *TCP*.
   iii. In the *Interval* field, enter 10.
   iv. In the *Timeout* field, enter 2.
   v. In the *Retry* field, enter 3.
   vi. In the *Port* field, enter 8013. Click *OK*.
b. Create a virtual server:
   i. Go to *Policy & Objects* and create a virtual server.
   ii. Configure the fields as follows:

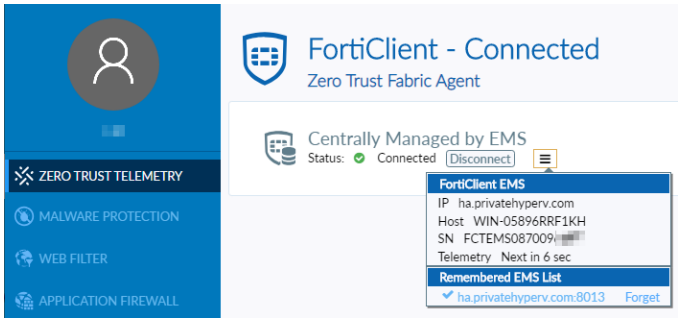
| Field | Value |
|---|---|
| *Virtual server IP* | VIP that you configured in step 4.f. In this example, the VIP is 172.16.1.50. |
| *Virtual server port* | 10443 |
| *Load Balancing method* | *First Alive* |
| *Health check* | Monitor that you configured. |
| *Network Type* | *TCP* |

   iii. Under *Real Servers*, select *Create New*.
   iv. In the *IPv4 address* field, enter the primary EMS node IP address. In this example, it is 192.168.1.10.
   v. In the *Port* field, enter 10443.
   vi. In the *Max connections* field, enter 0.
   vii. For *Mode*, select *Active*.
   viii. Create a real server for the secondary EMS node. Click *Save*.
c. Repeat steps i-ix to create five additional virtual servers. The additional servers use ports 443, 8013, 8015, 8443, and 8871, but otherwise have identical settings to the first virtual server created. If you have enabled Chromebook management, create a virtual server for port 8443. Similarly, if you require importing an ACME certificate, create a virtual server for port 80.
d. Create a security policy that includes the LB virtual server as a destination address:
   i. Go to *Policy & Objects > Firewall Policy*.
   ii. Click *Create New*.
   iii. Configure the *Incoming Interface* and *Outgoing Interface* fields. The outgoing interface connects to the primary EMS node.
   iv. For *Source*, select *all*.
   v. In the *Destination* field, select ports 10443, 443, 8013, 8015, 8443, and 8871.
   vi. For *Service*, select *ALL*.
   vii. For *Inspection Mode*, select *Proxy-based*.
   viii. Save the policy.
   ix. If the EMS nodes are in different subnets, repeat these steps to configure a policy for the secondary EMS node. In this example, the nodes are in the same subnet, so you do not need to add a separate policy for the secondary EMS.

3. After the FortiGate LB configuration is complete, you can access EMS using the VIP configured in the FortiGate LB. If after initially installing EMS 7.4.4 you need to upgrade to a newer build, repeat steps 4.a.-c. with the new installation file.

**To validate the EMS HA configuration:**

1. Go to *Deployment & Installers* > *Manage Deployment*. Create a deployment package to deploy FortiClient to endpoints. See Adding a FortiClient installer.
2. On an endpoint, download the deployment package from the download link.
3. Install FortiClient on the endpoint.
4. Ensure that FortiClient can register to the EMS server successfully using the FQDN.
5. Simulate HA by stopping *FortiClient Endpoint Management Server Monitor Service* on the primary EMS node. Ensure that the secondary EMS node is now the primary.
6. Ensure that FortiClient can still register to the EMS successfully using the FQDN.



www.fortinet.com