



**Postgres Enterprise Manager
Planning Guide**
Release 8.6.0

EnterpriseDB Corporation

Oct 28, 2022

CONTENTS

1	Understanding Postgres Enterprise Manager components and architecture	3
1.1	PEM architecture	4
1.1.1	PEM server	4
1.1.2	PEM agent	6
1.1.3	PEM web client	7
1.1.4	SQL Profiler plugin	8
2	Deployment considerations	9
2.1	Web server installation options	9
2.1.1	PEM server and Apache web server on separate hosts	9
3	Deployment considerations	11
3.1	Using Failover Manager for high availability	11
3.1.1	Initial product installation and configuration	12
3.1.2	Set up the primary node for streaming replication	12
3.1.3	Set up the standby nodes for streaming replication	13
3.1.4	Set up EFM to manage failover on all hosts	15
3.1.5	Current limitations	20
3.2	Connection pooling using pgBouncer	20
3.2.1	PEM server and agent connection management mechanism	20
3.2.2	Preparing the PEM database server	23
3.2.3	Configuring PgBouncer	24
3.2.4	Configuring the PEM agent	26
3.3	Authentication options	28
3.3.1	Configuring the PEM server to use Windows Active Directory domain services for Kerberos authentication (SSPI)	28
3.3.2	Configuring the PEM server to use Kerberos authentication	35
3.3.3	Configuring the PEM server to use two-factor authentication	39
3.4	Securing your deployment	40
3.4.1	Apache HTTPD security configurations	41
3.4.2	PEM application security configurations	46
3.5	Web server installation options	49
3.5.1	PEM server and Apache web server on separate hosts	49

Welcome to Postgres Enterprise Manager (PEM) Planning Guide. PEM consists of components that provide the management and analytical functionality for your EDB Postgres Advanced Server or PostgreSQL database. PEM is based on the Open Source pgAdmin 4 project.

PEM is a comprehensive database design and management system. PEM is designed to meet the needs of both novice and experienced Postgres users alike, providing a powerful graphical interface that simplifies the creation, maintenance, and use of database objects.

UNDERSTANDING POSTGRES ENTERPRISE MANAGER COMPONENTS AND ARCHITECTURE

Postgres Enterprise Manager (PEM) monitors and manages multiple Postgres servers through a single graphical interface. PEM can monitor the following areas of the infrastructure:

- ****Hosts**** — One **or** more servers (physical **or** virtual) **and** their operating systems.
- ****Databaseservers**** — One **or** more instances of PostgreSQL **or** EDB Postgres Advanced Server **or** EDB Postgres Extended Server (formerly known as 2ndQPostgres) running on a host.
- ****Databases**** — One **or** more databases **and** their schema objects, such as tables **and** indexes.

Note: The term Postgres refers to PostgreSQL, EDB Postgres Advanced Server, or EDB Postgres Extended Server.

PEM consists of individual software components:

- ****PEMserver**** — The PEM server **is** the data repository **for** monitoring data **and** a server to which agents **and** clients connect. The PEM server consists of an instance of PostgreSQL, an associated database **for** storing monitoring data, **and** a server that provides web services.
- ****PEMagent**** — The PEM agent **is** responsible **for** executing tasks **and** reporting statistics **from the** agent host **and** the monitored Postgres instances to the PEM server. A single PEM agent can monitor multiple installed instances of Postgres that reside on one **or** many hosts.
- ****PEMwebclient**** — The PEM web interface allows you to manage **and** monitor Postgres servers **and** use PEM extended functionality. The web interface software **is** installed **with** the PEM server **and is** accessed using **any** supported web browser.
- ****SQLProfiler**** — SQL Profiler is a Postgres server plugin to record the monitoring data and query plans for the SQL Profiler tool to analyze in PEM. This is an optional component of PEM, but the plugin must be installed in each instance of Postgres for which you want to use it. You can use the SQL Profiler with any supported version of an EDB distribution of a PostgreSQL server or EDB Postgres Advanced Server, not just those managed through the PEM server. See [:ref:`SQL Profiler Configuration </pe/latest/profiling\\_workloads/pe-sqlprofiler/`](https://www.enterprisedb.com/SQL-Profiler-Configuration/pe/latest/profiling_workloads/pe-sqlprofiler/) for details and supported versions. (continues on next page)

(continued from previous page)

1.1 PEM architecture

The following architectural diagram shows the relationships between the PEM server, clients, and managed and un-managed Postgres servers.

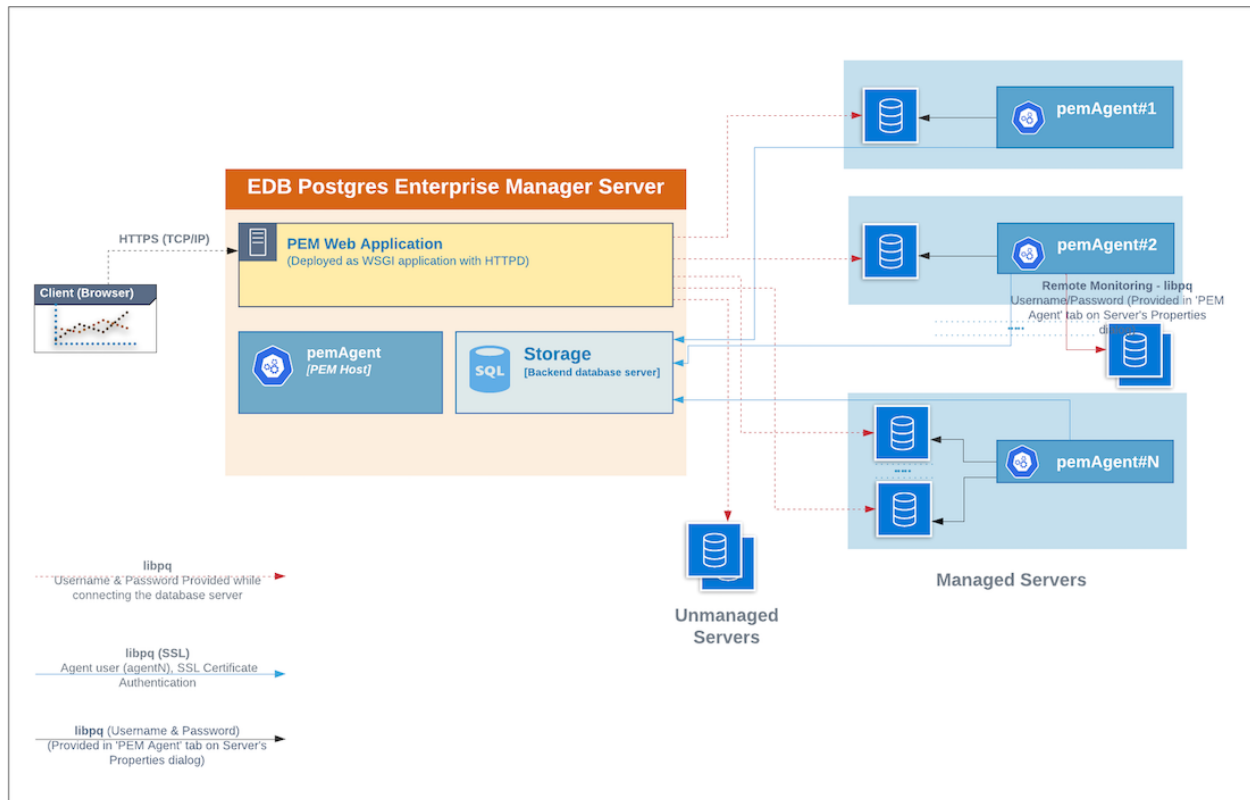


Fig. 1: PEM Architecture

1.1.1 PEM server

The PEM server consists of an instance of Postgres, an instance of the Apache web server providing web services to the client, and a PEM Agent. PEM uses a server-side cryptographic plugin to generate authentication certificates.

The instance of Postgres (a database server) and an instance of the Apache web-server HTTPD) can be on the same host or on separate hosts.

Note: All the PEM features are available on either backend database server you select: PostgreSQL or EDB Postgres Advanced Server.

- ****Postgresinstance(databaseserver)**** — This is the backend database server.
 - ↳ It hosts a database named `pem`, which acts as the repository for PEM server.
 - ↳ The `pem` database contains several schemas that store metric data collected from each monitored host, server, and database.

(continues on next page)

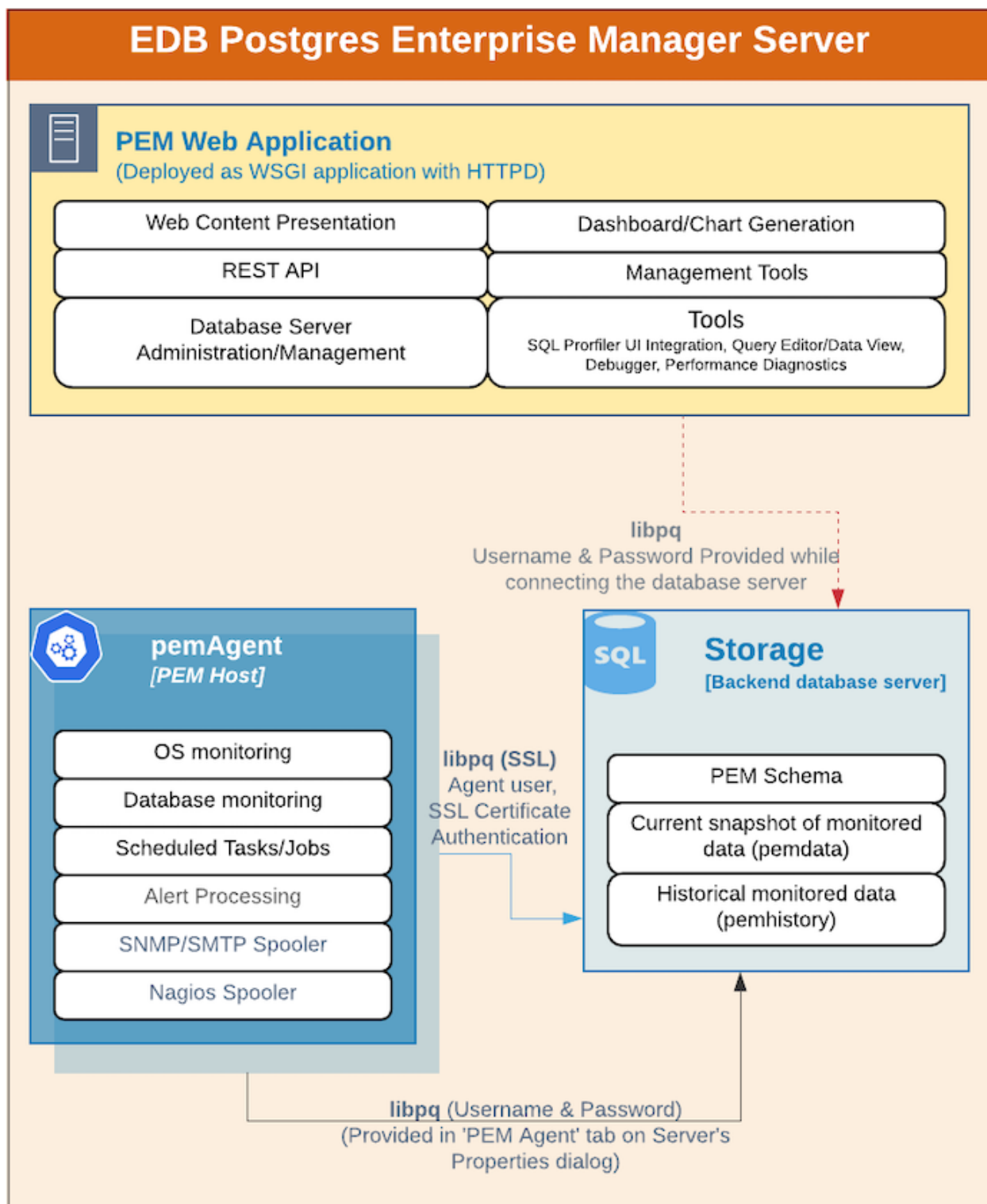


Fig. 2: PEM Server

(continued from previous page)

- pem — This schema is the core of the PEM application. It contains the definitions of configuration functions, tables, or views required by the application.
 - pemdata — This schema stores the current snapshot of the monitored data.
 - pemhistory — This schema stores the historical monitored data.
- **Apachewebserver(HTTPD)** — The PEM web application is deployed as a WSGI application with HTTPD to provide web services to the client. It is made up of the following:
 - **Webcontentpresentation** — The presentation layer is created by the web application (such as browser and login page).
 - **RestAPI** — The REST API allows integration with other apps and services.
 - **Databaseserveradministration/management** — You can perform database server administration and management activities like CREATE, ALTER, and DROP for managed and unmanaged servers.
 - **Dashboard/chartgeneration** — Internally, the web application includes functionality that generates dashboards and charts.
 - **Managementtools** — The Audit Manager, Capacity Manager, Log Manager, Postgres Expert, Postgres Log Analysis Expert, and the Tuning Wizard are available in the web application.
 - Other tools provide functionality on managed or unmanaged servers:
 - **SQLProfilerUIintegration** — SQL Profiler generates easily analyzed traces of session content.
 - **Queryeditor/dataview** — The Query editor allows you to query, edit, and view data.
 - **Debugger** — The debugger helps you debug queries.
 - **Performancediagnostics** — Performance diagnostics help you analyze the performance of Postgres instances.

We recommend that you use a dedicated machine to host production instances of the PEM backend database. The host might be subject to high levels of data throughput, depending on the number of database servers that are being monitored and the workloads the servers are processing.

1.1.2 PEM agent

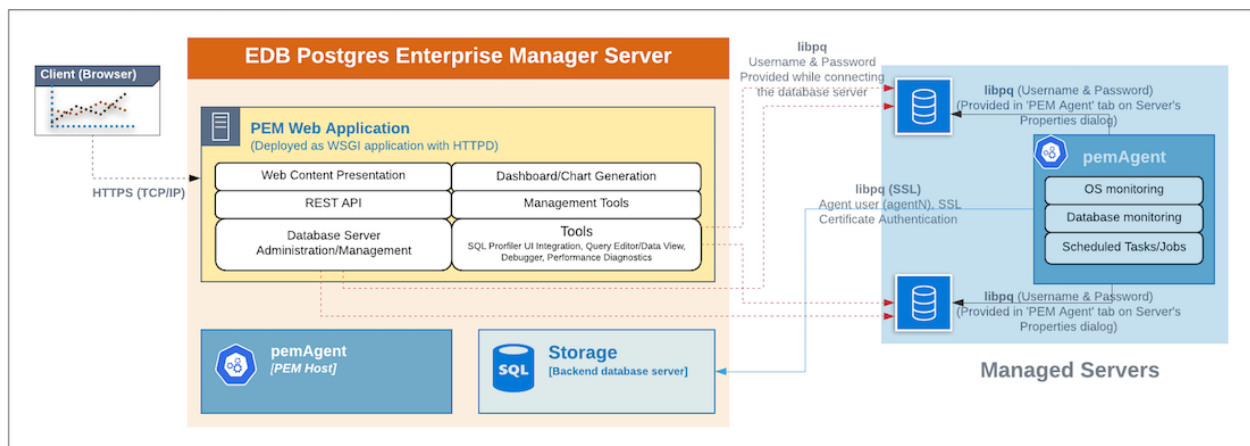


Fig. 3: PEM Agent

The PEM agent is responsible for collecting monitoring data from the machine and operating system and from each of the Postgres instances to which they are bound. Each PEM agent can monitor one physical or virtual machine and is capable of monitoring multiple database servers locally that are installed on other systems. These servers can be

installed on the same system or remotely. It's also responsible for executing other tasks that the user might schedule such as server shutdowns, SQL Profiler traces, and custom jobs.

A PEM agent is installed by default on the PEM server when you install the PEM server. It is generally referred to as a PEM agent on the PEM host. Separately, you can also install the PEM agent on the other servers hosting the Postgres instances you want to monitor using PEM.

Whether monitoring locally or remotely, the PEM agent connects to the PEM server using PostgreSQL's libpq, using SSL certificate-based authentication. The PEM agent installer in Windows and pemworker CLI in Linux is responsible for registering each agent with the PEM server and generating and installing the required certificates.

There is only one-way traffic between the PEM agent and PEM server. The PEM agent always connects to the PEM server.

The PEM agent must be able to connect to each database server that it monitors. This connection is made over a TCP/IP connection (or, optionally, a Unix Domain Socket on Unix hosts), and can optionally use SSL. You must configure the connection and authentication to the monitored server.

Once configured, each agent collects statistics and other information on the host and each database server and database that it monitors. Each piece of information is known as a *metric* and is collected by a *probe*. Most probes collect multiple metrics at once for efficiency. Examples of the metrics collected include:

- Disk I/O statistics
- Network statistics
- Database server version string
- Database server configuration option (GUC) values
- Table access statistics
- Table and index sizes

For a list of PEM probes, see [Probes](#) .

By default, the PEM agent bound to the database server collects the OS/database monitoring statistics and also runs any scheduled tasks/jobs for that particular database server, storing data in the `pem` database on the PEM server.

The alert processing, SNMP/SMTP spoolers, and Nagios spooler data is stored in the `pem` database on the PEM server and is then processed by the PEM agent on the PEM host by default. However, you can enable processing by other PEM Agents by adjusting the SNMP/SMTP and Nagios parameters of the PEM agents.

For more information about these parameters, see [Mail server configuration](#) .

1.1.3 PEM web client

The PEM client is a web-based application that runs in supported browsers. The client's web interface connects to the PEM server and allows direct management of managed or unmanaged servers and the databases and schemas that reside on them.

The client allows you to use PEM functionality that makes use of the data logged on the server through features such as dashboards, the Postgres Log Analysis Expert, and Capacity Manager.

1.1.4 SQL Profiler plugin

You don't have to install the SQL Profiler plugin on every server, but you must install and configure the plugin on each server on which you want to use the SQL Profiler. You might also want to install and configure SQL Profiler on unmonitored development servers. You can also temporarily install the SQL Profiler plugin for ad hoc use.

The plugin is installed with the EDB Postgres Advanced Server distribution but must be installed separately for use with PostgreSQL. The SQL Profiler installer is available from the [EDB website](#).

You can use SQL Profiler on servers that aren't managed through PEM. However, to perform scheduled traces, a server must have the plugin installed and must be managed by an installed and configured PEM agent.

For more information about using SQL Profiler, see *[SQL Profiler plugin](#)*.

DEPLOYMENT CONSIDERATIONS

Consider these issues before deploying Postgres Enterprise Manager.

Considerations	Implementation instructions
Is a standalone server enough or do you need a high availability architecture?	Installing the server or Deploying high availability
Do you need to implement connection pooling?	Deploying connection pooling
What type of authentication do you want to use?	<i>Authentication options</i>
What actions should you take to avoid security vulnerabilities?	<i>Securing your deployment</i>
Where should you host the Apache web server?	<i>Web server installation options</i>

2.1 Web server installation options

During the PEM server installation, you can specify your hosting preferences for the Apache web server.

2.1.1 PEM server and Apache web server on separate hosts

1. **Install the PEM server on both the hosts. For steps, see**
Installing the PEM server on Linux .
2. Configure the PEM server host by selecting the **Database** option on the first host.
3. Configure an Apache web server by selecting the **WebServices** option on the second host.

For detailed information about configuring a PEM server, see
Configuring the PEM server on Linux platforms .

Note: For production environments, best practice is to have PEM server and Apache web server on separate hosts.
PEM server and Apache web server on the same host

1. Install the PEM server. For steps, see Installing the PEM server on Linux .

2. Run the configuration script. Select the **WebServicesandDatabase** option to install PEM server and Apache web server on the same host. For steps, see [Configuring the PEM server on Linux platforms](#) .

Note: For nonproduction environments, a best practice is to have PEM server and Apache web server on the same hosts.

DEPLOYMENT CONSIDERATIONS

There are a number of things to consider before deploying Postgres Enterprise Manager.

Considerations	Implementation instructions
Is a standalone server sufficient or do you need a high availability architecture?	Installing the server or Deploying high availability
Do you need to implement connection pooling?	Deploying connection pooling
What type of authentication to use?	<i>Authentication options</i>
What actions should you take to avoid security vulnerabilities?	<i>Securing your deployment</i>
Where to host the Apache web server?	<i>Web server installation options</i>

3.1 Using Failover Manager for high availability

Postgres Enterprise Manager (PEM) helps database administrators, system architects, and performance analysts to administer, monitor, and tune Postgres database servers.

Failover Manager is a high-availability tool from EDB that enables a Postgres primary node to failover to a standby node during a software or hardware failure on the primary.

The examples in the following sections use these IP addresses:

- 172.16.161.200 - PEM Primary
- 172.16.161.201 - PEM Standby 1
- 172.16.161.202 - PEM Standby 2
- 172.16.161.203 - EFM Witness Node
- 172.16.161.245 - PEM VIP (used by agents and users to connect)

The following must use the VIP address:

- The PEM agent binding of the monitored database servers
- Accessing the PEM web client
- Accessing the webserver services

3.1.1 Initial product installation and configuration

1. Install the following on the primary and one or more standbys:

- EDB Postgres Advanced Server (backend database for PEM Server) - *PEM server* - EDB Failover Manager 4.1

Refer to the installation instructions in the product documentation using these links or see the instructions on the [EDB repos website](#) . Replace USERNAME:PASSWORD with your username and password in the instructions to access the EDB repositories.

Make sure that the database server is configured to use the scram-sha-256 authentication method, as the PEM server configuration script doesn't work with trust authentication.

You must install the `java-1.8.0-openjdk` package to install EFM.

2. Configure the PEM server on the primary server with an initial configuration of type 1 (web services and database):

edb_notranlate_0 For more detail on configuration types see,
Configuring the PEM server on Linux .

3. Add the following ports in the firewall on the primary and all the standby servers to allow the access:

- 8443 for PEM Server (https) - 5444 for EPAS 13 - 7800 for EFM - 7908 for EFM Admin

For example:

```
$ sudo firewall-cmd --zone=public --add-port=5444/tcp --permanent
success
$ sudo firewall-cmd --zone=public --add-port=8443/tcp --permanent
success
$ sudo firewall-cmd --zone=public --add-port=7800/tcp --permanent
success
$ sudo firewall-cmd --zone=public --add-port=7809/tcp --permanent
success
$ sudo firewall-cmd --reload
success
```

3.1.2 Set up the primary node for streaming replication

1. Create the replication role:

```
$ /usr/edb/as13/bin/psql -h 172.16.161.200 -p 5444 -U enterprisedb edb -c "CREATE ROLE
↳ repl REPLICATION LOGIN PASSWORD password";
CREATE ROLE
```

Give the password of your choice.

2. Configure the following in the `postgresql.conf` file:

```
wal_level = replica
max_wal_senders = 10
wal_keep_size = 500
max_replication_slots = 10
```

For more information on configuring parameters for streaming replication, see the [PostgreSQL documentation](#) .

Note: The configuration parameters might differ for different versions of the database server. You can email EDB Support at techsupport@enterprisedb.com for help with setting up these parameters.

3. Add the following entry in the host-based authentication (`/var/lib/edb/as13/data/pg_hba.conf`) file to allow the replication user to connect from all the standbys:

```
hostssl replication repl 172.16.161.201/24 scram-sha-256
```

Note: You can change the cidr range of the IP address, if needed.

4. Modify the host-based authentication (`/var/lib/edb/as13/data/pg_hba.conf`) file for the `pem_user` role to connect to all databases using the `scram-sha-256` authentication method:

```
# Allow local PEM agents and admins to connect to PEM server
hostssl all +pem_user 172.16.161.201/24 scram-sha-256
hostssl pem +pem_user 127.0.0.1/32 scram-sha-256
hostssl pem +pem_agent 127.0.0.1/32 cert
# Allow remote PEM agents and users to connect to PEM server
hostssl pem +pem_user 0.0.0.0/0 scram-sha-256
hostssl pem +pem_agent 0.0.0.0/0 cert
```

5. Restart the EPAS 13 server.

```
systemctl restart edb-as-13.service
```

3.1.3 Set up the standby nodes for streaming replication

1. Stop the service for EPAS 13 on all the standby nodes:

```
$ systemctl stop edb-as-13.service
```

Note: This example uses the `pg_basebackup` utility to create the replicas of the PEM backend database server on the standby servers. When using `pg_basebackup`, you need to stop the existing database server and remove the existing data directories.

2. Remove the data directory of the database server on all the standby nodes:

```
$ sudo su - enterprisedb
```

```
$ rm -rf /var/lib/edb/as13/data/*
```

3. Create the `.pgpass` file in the home directory of the `enterprisedb` user on all the standby nodes:

```
$ sudo su - enterprisedb
```

```
$ cat > ~/.pgpass << _EOF_
172.16.161.200:5444:replication:repl:CHANGE_ME
172.16.161.201:5444:replication:repl:CHANGE_ME
```

(continues on next page)

(continued from previous page)

```
172.16.161.202:5444:replication:repl:CHANGE_ME
_EOF_

$ chmod 600 ~/.pgpass
```

4. Take the backup of the primary node on each of the standby nodes using pg_basebackup:

```
$ sudo su - enterprisedb /usr/edb/as13/bin/pg_basebackup -h 172.16.161.200 \
-D /var/lib/edb/as13/data -U repl -v -P -Fp -R -p 5444
```

The backup command creates the postgresql.auto.conf and standby.signal files on the standby nodes. The postgresql.auto.conf file has the following content:

```
sudo su - enterprisedb cat /var/lib/edb/as13/data/postgresql.auto.conf
# Do not edit this file manually
# It will be overwritten by the ALTER SYSTEM command.
primary_conninfo = 'user=repl passfile="/var/lib/edb/.pgpass" channel_binding=prefer_
→host=172.16.161.200 port=5444 sslmode=prefer sslcompression=0 ssl_min_protocol_
→version=TLSv1.2 gssencmode=prefer krbsvrname=postgres target_session_attrs=any'
```

5. In the postgresql.conf file on each of the standby nodes, edit the following parameter:

```
hot_standby = on
```

6. Start the EPAS 13 database server on each of the standby nodes:

```
$ systemctl enable edb-as-13

$ systemctl start edb-as-13
```

7. Copy the following files from the primary node to the standby nodes at the same location, overwriting any existing files. Set the permissions on the files:

- /etc/httpd/conf.d/edb-pem.conf - /etc/httpd/conf.d/edb-ssl-pem.conf - /root/.pem/agent1.crt - /root/.pem/agent1.key - /usr/edb/pem/agent/etc/agent.cfg - /usr/edb/pem/share/.install-config - /usr/edb/pem/web/pem.wsgi - /usr/edb/pem/web/config_setup.py

For example:

```
$ mkdir -p /root/.pem
$ chown root:root /root/.pem
$ chmod 0755 /root/.pem
$ mkdir -p /var/lib/pemhome/.pem
$ chown pem:pem /var/lib/pemhome/.pem
$ chmod 0700 /var/lib/pemhome/.pem
$ mkdir -p /usr/edb/pem/logs
$ chown root:root /usr/edb/pem/logs
$ chmod 0755 /usr/edb/pem/logs
$ for file in /etc/httpd/conf.d/edb-pem.conf \
              /etc/httpd/conf.d/edb-ssl-pem.conf \
              /root/.pem/agent1.crt \
              /usr/edb/pem/agent/etc/agent.cfg \
              /usr/edb/pem/share/.install-config \
```

(continues on next page)

(continued from previous page)

```

        /usr/edb/pem/web/pem.wsgi \
        /usr/edb/pem/web/config_setup.py; do \
    chown root:root ${file}; \
    chmod 0644      ${file}; \
done;
$ chmod 0600 /root/.pem/agent1.key
$ chown root:root /root/.pem/agent1.key

```

This code ensures that the webserver is configured on the standby and is disabled by default. Switchover by EFM enables the webserver.

Note: Manually keep the certificates in sync on master and standbys whenever the certificates are updated.

8. Run the `configure-selinux.sh` script to configure the SELinux policy for PEM:

```

$ /usr/edb/pem/bin/configure-selinux.sh
getenforce found, now executing getenforce command
Configure the httpd to work with the SELinux
Allow the httpd to connect the database (httpd_can_network_connect_db = on)
Allow the httpd to connect the network (httpd_can_network_connect = on)
Allow the httpd to work with cgi (httpd_enable_cgi = on)
Allow to read & write permission on the pem user home directory
SELinux policy is configured for PEM

$ sudo chmod 640 /root/.pem/agent1.crt

```

9. Disable and stop HTTPD and PEM agent services if they're running on all replica nodes:

```

systemctl stop pemagent
systemctl stop httpd
systemctl disable pemagent
systemctl disable httpd

```

Note: At this point, a PEM primary server and two standbys are ready to take over from the primary whenever needed.

3.1.4 Set up EFM to manage failover on all hosts

1. Prepare the primary node to support EFM:

- Create a database user `efm` to connect to the database servers. - Grant the execute privileges on the functions related to WAL logs and the monitoring privileges to the user. - Add entries in `pg_hba.conf` to allow the `efm` database user to connect to the database server from all nodes on all the hosts. - Reload the configurations on all the database servers. For example:

```

$ cat > /tmp/efm-role.sql << _EOF_
-- Create a role for EFM
CREATE ROLE efm LOGIN PASSWORD password;

-- Give privilege to efm user to connect to a database

```

(continues on next page)

(continued from previous page)

```

GRANT CONNECT ON DATABASE edb TO efm;

-- Give privilege to efm user to do backup operations
GRANT EXECUTE ON FUNCTION pg_current_wal_lsn() TO efm;
GRANT EXECUTE ON FUNCTION pg_last_wal_replay_lsn() TO efm;
GRANT EXECUTE ON FUNCTION pg_wal_replay_resume() TO efm;
GRANT EXECUTE ON FUNCTION pg_wal_replay_pause() TO efm;
GRANT EXECUTE ON FUNCTION pg_reload_conf() TO efm;

-- Grant monitoring privilege to the efm user
GRANT pg_monitor TO efm;
_EOF_

$ /usr/edb/as13/bin/psql -h 172.16.161.200 -p 5444 -U enterprisedb edb -f /tmp/efm-role.
↪sql
CREATE ROLE
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT ROLE

$ rm -f /tmp/efm-role.sql

$ cat > /var/lib/edb/as13/data/pg_hba.conf <<< _EOF_
hostssl      edb      efm      172.16.161.200/32      scram-sha-256
hostssl      edb      efm      172.16.161.201/32      scram-sha-256
hostssl      edb      efm      172.16.161.202/32      scram-sha-256
hostssl      edb      efm      172.16.161.203/32      scram-sha-256
_EOF_

$ /usr/edb/as13/bin/psql -h 172.16.161.200 -p 5444 -U enterprisedb edb -c "SELECT pg_
↪reload_conf();"

```

2. Create the scripts on each node to start/stop the PEM agent:

```

$ sudo cat > /usr/local/bin/start-httpd-pemagent.sh << _EOF_
#!/bin/sh
/bin/sudo /bin/systemctl enable httpd
/bin/sudo /bin/systemctl start httpd
/bin/sudo /bin/systemctl enable pemagent
/bin/sudo /bin/systemctl start pemagent
_EOF_
$ sudo cat > /usr/local/bin/stop-httpd-pemagent.sh << _EOF_
#!/bin/sh

/bin/sudo /bin/systemctl stop pemagent
/bin/sudo /bin/systemctl disable pemagent
/bin/sudo /bin/systemctl stop httpd
/bin/sudo /bin/systemctl disable httpd

```

(continues on next page)

(continued from previous page)

```
_EOF_
$ sudo chmod 770 /usr/local/bin/start-pemagent.sh
$ sudo chmod 770 /usr/local/bin/stop-pemagent.sh
```

3. Create a sudoers file (/etc/sudoers.d/efm-pem) on each node to allow the efm user to start/stop the pemagent:

```
$ sudo cat > /etc/sudoers.d/efm-pem << _EOF_
efm    ALL=(ALL)          NOPASSWD:    /bin/systemctl enable pemagent
efm    ALL=(ALL)          NOPASSWD:    /bin/systemctl disable pemagent
efm    ALL=(ALL)          NOPASSWD:    /bin/systemctl stop pemagent
efm    ALL=(ALL)          NOPASSWD:    /bin/systemctl start pemagent
efm    ALL=(ALL)          NOPASSWD:    /bin/systemctl status pemagent
_EOF_
```

4. Create an efm.nodes file on all nodes using the sample file (/etc/edb/efm-4.1/efm.nodes.in), and give read-write access to the efm OS user:

```
$ sudo cp /etc/edb/efm-4.1/efm.nodes.in /etc/edb/efm-4.1/efm.nodes
$ sudo chown efm:efm /etc/edb/efm-4.1/efm.nodes
$ sudo chmod 600 /etc/edb/efm-4.1/efm.nodes
```

5. Add the IP address and efm port of the primary node in the /etc/edb/efm-4.1/efm.nodes file on the standby nodes:

```
$ sudo cat > /etc/edb/efm-4.1/efm.nodes <<< _EOF_
172.16.161.200:7800
_EOF_
```

6. Create the efm.properties file on all the nodes using the sample file (/etc/edb/efm-4.1/efm.properties.in). Grant read access to all the users:

```
$ sudo cp /etc/edb/efm-4.1/efm.properties.in /etc/edb/efm-4.1/efm.properties
$ sudo chown efm:efm /etc/edb/efm-4.1/efm.properties
$ sudo chmod a+r /etc/edb/efm-4.1/efm.properties
```

7. Encrypt the efm user's password using the efm utility:

```
$ export EFMPASS=password
$ /usr/edb/efm-4.1/bin/efm encrypt efm --from-env
096666746b05b081d1a98e43d94c9dad
```

8. Edit the following parameters in the properties file:

```
db.user=efm
db.password.encrypted=096666746b05b081d1a98e43d94c9dad
db.port=5444
db.database=edb
db.service.owner=enterprisedb
db.service.name=edb-as-13
db.bin=/usr/edb/as13/bin
db.data.dir=/var/lib/edb/as13/data
jdbc.sslmode=require
```

(continues on next page)

(continued from previous page)

```
user.email=username@example.com
from.email=node1@efm-pem
notification.level=INFO
notification.text.prefix=[PEM/EFM]
bind.address=172.16.161.200:7800
admin.port=7809
is.witness=false
local.period=10
local.timeout=60
local.timeout.final=10
remote.timeout=10
node.timeout=50
encrypt.agent.messages=true
stop.isolated.primary=true
stop.failed.primary=true
primary.shutdown.as.failure=false
update.physical.slots.period=0
ping.server.ip=8.8.8.8
ping.server.command=/bin/ping -q -c3 -w5
auto.allow.hosts=false
stable.nodes.file=false
db.reuse.connection.count=0
auto.failover=true
auto.reconfigure=true
promotable=true
use.replay.tiebreaker=true
standby.restart.delay=0
reconfigure.num.sync=false
reconfigure.sync.primary=false
minimum.standbys=0
recovery.check.period=1
restart.connection.timeout=60
auto.resume.period=0
virtual.ip=172.16.161.245
virtual.ip.interface=ens33
virtual.ip.prefix=24
virtual.ip.single=true
check.vip.before.promotion=true
pgpool.enable=false
sudo.command=sudo
sudo.user.command=sudo -u %u
syslog.host=localhost
syslog.port=514
syslog.protocol=UDP
syslog.facility=LOCAL1
file.log.enabled=true
syslog.enabled=false
jgroups.loglevel=INFO
efm.loglevel=INFO
jvm.options=-Xmx128m
script.remote.post.promotion=/usr/local/bin/stop-pemagent.sh
script.post.promotion=/usr/local/bin/start-pemagent.sh
```

9. Set the value of the `is.witness` configuration parameter on the witness node to `true` :

```
is.witness=true
```

10. Enable and start the EFM service on the primary node:

```
$ systemctl enable edb-efm-4.1
$ systemctl start edb-efm-4.1
```

11. Allow the standbys to join the cluster started on the primary node:

```
/usr/edb/efm-4.1/bin/efm allow-node efm 172.16.161.201
/usr/edb/efm-4.1/bin/efm allow-node efm 172.16.161.202
/usr/edb/efm-4.1/bin/efm allow-node efm 172.16.161.203
```

12. Enable and start the EFM service on the standby nodes and the EFM witness node:

```
$ systemctl enable edb-efm-4.1
$ systemctl start edb-efm-4.1
```

13. Check the EFM cluster status from any node:

```
$ sudo /usr/edb/efm-4.1/bin/efm cluster-status efm
Cluster Status: efm
  Agent Type  Address          DB      VIP
-----
Primary      172.16.161.200  UP      172.16.161.245*
Standby      172.16.161.201  UP      172.16.161.245
Standby      172.16.161.202  UP      172.16.161.245
Witness      172.16.161.203  N/A     172.16.161.245

Allowed node host list:
  172.16.161.200 172.16.161.201 172.16.161.202 172.16.161.203

Membership coordinator: 172.16.161.200

Standby priority host list:
  172.16.161.201 172.16.161.202

Promote Status:

  DB Type  Address          WAL Received LSN  WAL Replayed LSN  Info
-----
Primary   172.16.161.200
Standby   172.16.161.201  0/F7A3808         0/F7A3808
Standby   172.16.161.202  0/F7A3808         0/F7A3808

Standby database(s) in sync with primary. It is safe to promote.
```

This status confirms that EFM is set up successfully and managing the failover for the PEM server.

In case of failover, any of the standbys are promoted as the primary node, and PEM agents connect to the new primary node. You can replace the failed primary node with a new standby using this procedure.

3.1.5 Current limitations

The current limitations include:

- Web console sessions for the users are lost during the switchover.
- Per-user settings set from the Preferences dialog box are lost, as they're stored in local configuration files on the file system.
- Background processes, started by the Backup, Restore, and Maintenance dialogs boxes, and their logs aren't shared between the systems. They are lost during switchover.

3.2 Connection pooling using pgBouncer

You can use pgBouncer as a connection pooler for limiting the number of connections from the PEM agent to the Postgres Enterprise Manager (PEM) server on non-Windows machine:

- Preparing the PEM Database Server provides information about preparing the PEM database server to be used with pgBouncer.
- Configuring pgBouncer provides detailed information about configuring pgBouncer to make it work with the PEM database server.
- Configuring the PEM agent provides detailed information about configuring a PEM agent to connect to pgBouncer.

For detailed information about using the PEM web interface, see the
Accessing the web interface .

3.2.1 PEM server and agent connection management mechanism

Each PEM agent connects to the PEM database server using the SSL certificates for each user. For example, an agent with ID#1 connects to the PEM database server using the agent1 user.

Prior to PEM version 7.5, the following limitations disallowed the use of the connection pooler between the PEM server and PEM agent:

- The PEM agent uses an SSL certificate to connect to the PEM database server.
- It uses an individual user identifier when connecting to the PEM database server.

EDB modified the PEM agent to allow the agent to use a common database user (instead of the dedicated agent users) to connect to the PEM database server.

We recommend using PgBouncer version 1.9.0 or later as the connection pooler. Versions 1.9.0 or later support cert authentication. PEM agents can connect to pgBouncer using SSL certificates.

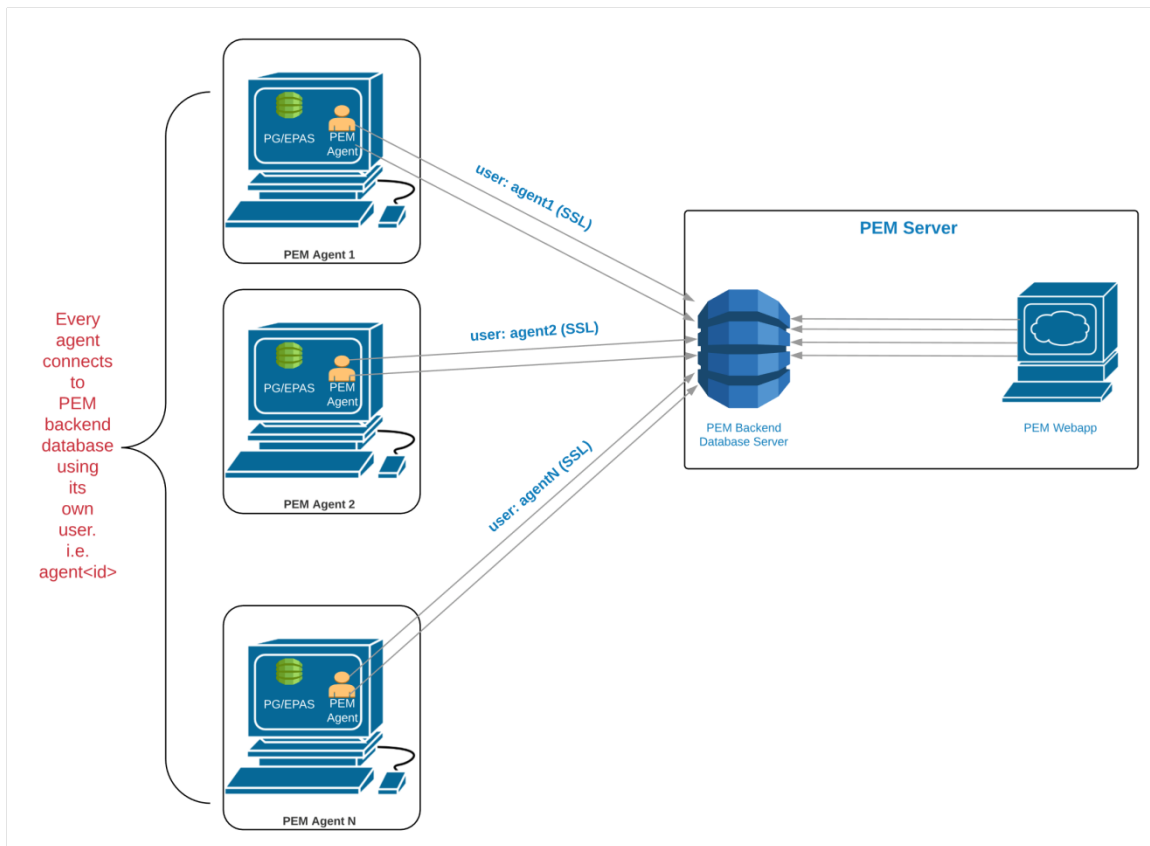


Fig. 1: Connecting to the PEM database without pgBouncer

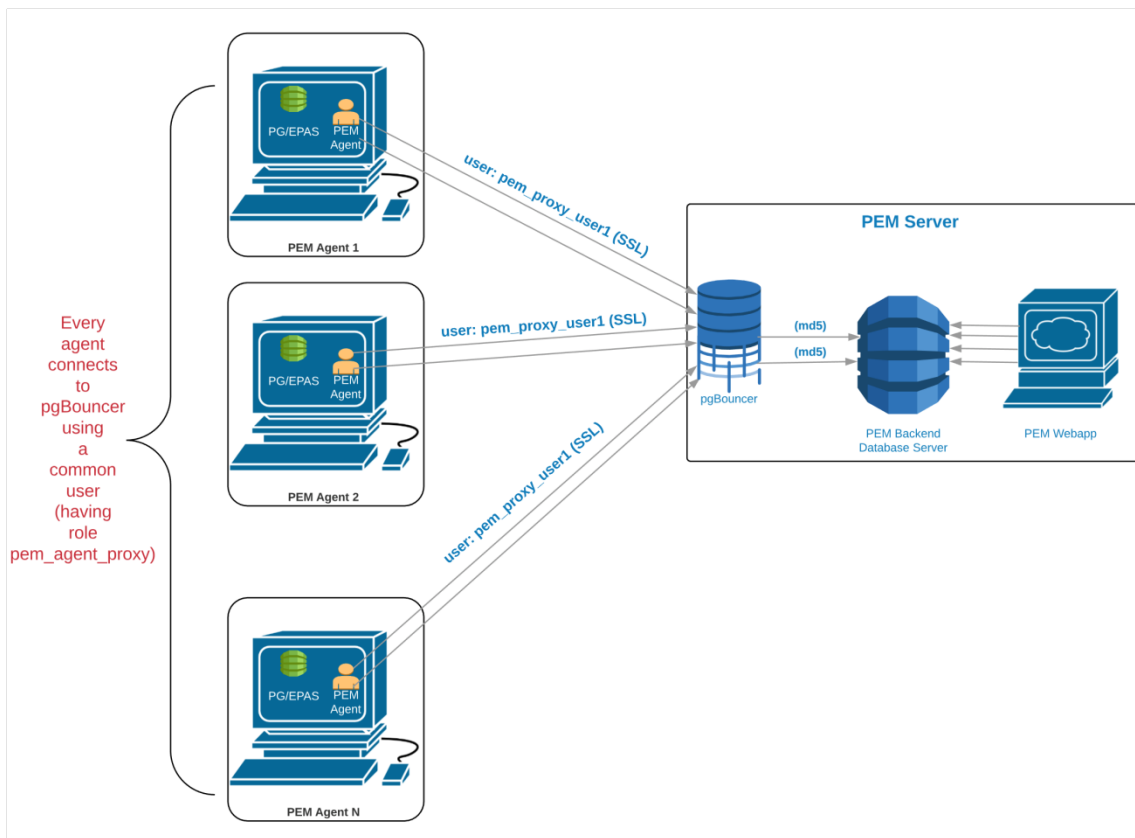


Fig. 2: Connecting to pgBouncer.

3.2.2 Preparing the PEM database server

You must configure the PEM database server to work with PgBouncer. This example shows how to configure the PEM database server.

1. Create a dedicated user named pgbouncer on the PEM database server:

```
pem=# CREATE USER pgbouncer PASSWORD ANY_PASSWORD LOGIN;
CREATE ROLE
```

2. Create a user named pem_admin1 (not a superuser) with pem_admin and pem_agent_pool role membership on the PEM database server:

```
pem=# CREATE USER pem_admin1 PASSWORD ANY_PASSWORD LOGIN
CREATE ROLE;
CREATE ROLE
pem=# GRANT pem_admin, pem_agent_pool TO pem_admin1;
GRANT ROLE
```

3. Grant CONNECT privileges to the pgbouncer user on the pem database:

```
pem=# GRANT CONNECT ON DATABASE pem TO pgbouncer ;GRANT USAGE ON
SCHEMA pem TO pgbouncer;
GRANT
```

4. Grant USAGE privileges to the pgbouncer user for the pem schema on the pem database:

```
pem=# GRANT USAGE ON SCHEMA pem TO pgbouncer;
GRANT
```

5. Grant EXECUTE privileges to the pgbouncer user on the pem.get_agent_pool_auth(text) function in the pem database. For example:

```
pem=# GRANT EXECUTE ON FUNCTION pem.get_agent_pool_auth(text) TO
pgbouncer;
GRANT
```

6. Use the pem.create_proxy_agent_user(varchar) function to create a user named pem_agent_user1 on the PEM database server:

```
pem=# SELECT pem.create_proxy_agent_user(pem_agent_user1);
__OUTPUT__
create_proxy_agent_user
-----
(1 row)
```

The function creates a user with the same name with a random password and grants pem_agent and pem_agent_pool roles to the user. This approach allows pgBouncer to use a proxy user on behalf of the agent.

7. Add the following entries to the start of the pg_hba.conf file of the PEM database server. These entries allow the pgBouncer user to connect to the pem database using the md5 authentication method.

```
# Allow the PEM agent proxy user (used by
# pgbouncer) to connect the to PEM server using
# md5
```

(continues on next page)

(continued from previous page)

```
local pem pgbouncer,pem_admin1 md5
```

3.2.3 Configuring PgBouncer

You must configure PgBouncer to work with the PEM database server. This example runs PgBouncer as the `enterprisedb` system user and outlines the process of configuring pgBouncer version 1.9 or later.

1. Open a terminal window and navigate to the pgBouncer directory.
2. Change the owner of the `etc` directory for pgBouncer (where `pgbouncer.ini` resides) to `enterprisedb`, and change the directory permissions to `0700`:

```
$ chown enterprisedb:enterprisedb /etc/edb/pgbouncer1.9
$ chmod 0700 /etc/edb/pgbouncer1.9
```

3. Change the contents of the `pgbouncer.ini` or `edb-pgbouncer.ini` file:

```
[databases]
;; Change the pool_size according to maximum connections allowed
;; to the PEM database server as required.
;; auth_user will be used for authenticate the db user (proxy
;; agent user in our case)

pem = port=5444 host=/tmp dbname=pem auth_user=pgbouncer
pool_size=80 pool_mode=transaction
* = port=5444 host=/tmp dbname=pem auth_user=pgbouncer
pool_size=10

[pgbouncer]
logfile = /var/log/edb/pgbouncer1.9/edb-pgbouncer-1.9.log
pidfile = /var/run/edb/pgbouncer1.9/edb-pgbouncer-1.9.pid
listen_addr = *
;; Agent needs to use this port to connect the pem database now
listen_port = 6432
;; Require to support for the SSL Certificate authentications
;; for PEM Agents
client_tls_sslmode = require
;; These are the root.crt, server.key, server.crt files present
;; in the present under the data directory of the PEM database
;; server, used by the PEM Agents for connections.
client_tls_ca_file = /var/lib/edb/as11/data/root.crt
client_tls_key_file = /var/lib/edb/as11/data/server.key
client_tls_cert_file = /var/lib/edb/as11/data/server.crt
;; Use hba file for client connections
auth_type = hba
;; Authentication file, Reference:
;; https://pgbouncer.github.io/config.html#auth_file
auth_file = /etc/edb/pgbouncer1.9/userlist.txt
;; HBA file
auth_hba_file = /etc/edb/pgbouncer1.9/hba_file
;; Use pem.get_agent_pool_auth(TEXT) function to authenticate
```

(continues on next page)

(continued from previous page)

```
;; the db user (used as a proxy agent user).
auth_query = SELECT * FROM pem.get_agent_pool_auth($1)
;; DB User for administration of the pgbouncer
admin_users = pem_admin1
;; DB User for collecting the statistics of pgbouncer
stats_users = pem_admin1
server_reset_query = DISCARD ALL
;; Change based on the number of agents installed/required
max_client_conn = 500
;; Close server connection if its not been used in this time.
;; Allows to clean unnecessary connections from pool after peak.
server_idle_timeout = 60
```

4. Create and update the `/etc/edb/pgbouncer1.9/userlist.txt` authentication file for PgBouncer:

```
pem=# COPY (
SELECT pgbouncer::TEXT, pgbouncer_password
UNION ALL
SELECT pem_admin1::TEXT, pem_admin1_password
TO /etc/edb/pgbouncer1.9/userlist.txt
WITH (FORMAT CSV, DELIMITER , FORCE_QUOTE *);

COPY 2
```

Note: A superuser can't invoke the PEM authentication query function `pem.get_proxy_auth(text)`. If the `pem_admin` user is a superuser, you must add the password to the authentication file (`enterprisedb` in the example).

5. Create an HBA file (`/etc/edb/pgbouncer1.9/hba_file`) for PgBouncer that contains the following content:

```
# Use authentication method md5 for the local connections to
# connect pem database & pgbouncer (virtual) database.
local pgbouncer all md5
# Use authentication method md5 for the remote connections to
# connect to pgbouncer (virtual database) using enterprisedb
# user.

host pgbouncer,pem pem_admin1 0.0.0.0/0 md5
```

```
# Use authentication method cert for the TCP/IP connections to
# connect the pem database using pem_agent_user1

hostssl pem pem_agent_user1 0.0.0.0/0 cert
```

6. Change the owner of the HBA file (`/etc/edb/pgbouncer1.9/hba_file`) to `enterprisedb`, and change the directory permissions to `0600`:

```
$ chown enterprisedb:enterprisedb /etc/edb/pgbouncer1.9/hba_file
$ chmod 0600 /etc/edb/pgbouncer1.9/hba_file
```

7. Enable the PgBouncer service, and start the service:

```
$ systemctl enable edb-pgbouncer-1.9

Created symlink from
/etc/systemd/system/multi-user.target.wants/edb-pgbouncer-1.9.service
to /usr/lib/systemd/system/edb-pgbouncer-1.9.service.

$ systemctl start edb-pgbouncer-1.9
```

3.2.4 Configuring the PEM agent

You can use an RPM package to install a PEM agent. For detailed installation information, see [Installing agent on Linux](#).

Don't configure the PEM agent responsible for sending SNMP notifications with pgBouncer. For example, if the default PEM agent installed with PEM server is used for SNMP notifications, don't configure it with pgBouncer.

Configuring a new PEM agent (installed via RPM)

After using an RPM package to install the PEM agent, you must configure it to work against a particular PEM database server. Use the following command:

```
$ PGSSLMODE=require PEM_SERVER_PASSWORD=pem_admin1_password
/usr/edb/pem/agent/bin/pemworker --register-agent --pem-server
pem_agent_user1 --display-name "Agent Name"
```

Postgres Enterprise Manager Agent registered successfully!

In this command, the `--pem-agent-user` argument instructs the agent to create an SSL certificate and key pair for the `pem_agent_user1` database user in the `/root/.pem` directory.

For example:

```
/root/.pem/pem_agent_user1.crt

/root/.pem/pem_agent_user1.key
```

The PEM agent uses the keys to connect to the PEM database server as `pem_agent_user1`. It also creates an agent configuration file named `/usr/edb/pem/agent/etc/agent.cfg`.

A line mentioning the agent-user to use appears in the `agent.cfg` configuration file. For example:

```
$ cat /usr/edb/pem/agent/etc/agent.cfg
[PEM/agent]
pem_host=172.16.254.22
pem_port=6432
agent_id=12
agent_user=pem_agent_user1
agent_ssl_key=/root/.pem/pem_agent_user1.key
agent_ssl_cert=/root/.pem/pem_agent_user1.crt
log_level=warning
log_location=/var/log/pem/worker.log
agent_log_location=/var/log/pem/agent.log
long_wait=30
```

(continues on next page)

(continued from previous page)

```

short_wait=10
alert_threads=0
enable_smtp=false
enable_snmp=false
enable_webhook=false
max_webhook_retries=3
allow_server_restart=true
allow_package_management=false
allow_streaming_replication=false
max_connections=0
connect_timeout=-1
connection_lifetime=0
allow_batch_probes=false
heartbeat_connection=false

```

Configuring an existing PEM agent (installed via RPM)

If you're using an existing PEM agent, you can copy the SSL certificate and key files to the target machine and reuse the files. You must modify the files, adding a new parameter and replacing some parameters in the existing `agent.cfg` file.

Add a line to use `agent_user` as the agent:

```
agent_user=pem_agent_user1
```

Update the port to specify the pgBouncer port:

```
pem_port=6432
```

Update the certificate and key path locations:

```

agent_ssl_key=/root/.pem/pem_agent_user1.key
agent_ssl_cert=/root/.pem/pem_agent_user1.crt

```

As an alternative, you can run the agent self-registration script. However, that process creates a new agent id. If run the agent self-registration script, you must replace the new agent id with the existing id and disable the entry for the new agent id in the `pem.agent` table. For example:

```

pem=# UPDATE pem.agent SET active = false WHERE id = <new_agent_id>;
__OUTPUT__

UPDATE 1

```

Note: Keep a backup of the existing SSL certificate, key file, and agent configuration file.

3.3 Authentication options

PEM also supports Kerberos and 2FA authentication. For implementation instructions, see:

On Linux:

- *Configuring the PEM server to use Kerberos authentication*
- *Configuring the PEM server to use Windows Kerberos server*

On Linux and Windows:

- *Configuring the PEM server to use two-factor authentication*

3.3.1 Configuring the PEM server to use Windows Active Directory domain services for Kerberos authentication (SSPI)

The Windows Active Directory domain service works with hostnames and not with IP addresses. To use single sign-on in PEM Server using Active Directory domain services, configure the following machines with hostnames using the DNS:

- Windows server (domain controller)
- PEM server (PEM web server and PEM backend database server)
- Client machine

For example, if the realm on Windows Active Directory is `edbpem.internal`, then you can set the Windows server hostname to `Krb5server.edbpem.internal`, the PEM server hostname to `pem.edbpem.internal`, and the client's hostname to `pg12.edbpem.internal`.

1. Install Active Directory, the PEM server, and the PEM backend database server

Perform the following installations:

- Install Active Directory on the Windows server (domain controller) that functions as the authentication server. Also, configure the Active Directory domain services to use Kerberos authentication, and then start it.
- Install the PEM server on a separate Linux machine. For more information, see [Installing the PEM server on Linux](#).
- Install the PEM backend database (Postgres/EDB Postgres Advanced Server) on the same Linux machine as the PEM server or a different one. For more information, see the installation steps on the

[EDB Docs website](#).

2. Create users in Active Directory to map with service principals

Create users in Active Directory of the Windows server to map with the HTTP service principal for the PEM web application.

1. Open **ActiveDirectoryUsersandComputers><DOMAIN_NAME>>Users**. Right-click and select **New>User**.
1. Enter the user details.
2. Give the password and make sure to clear **Usermustchange password at next login**. Also select **User cannot change password** and **Password never expires**.

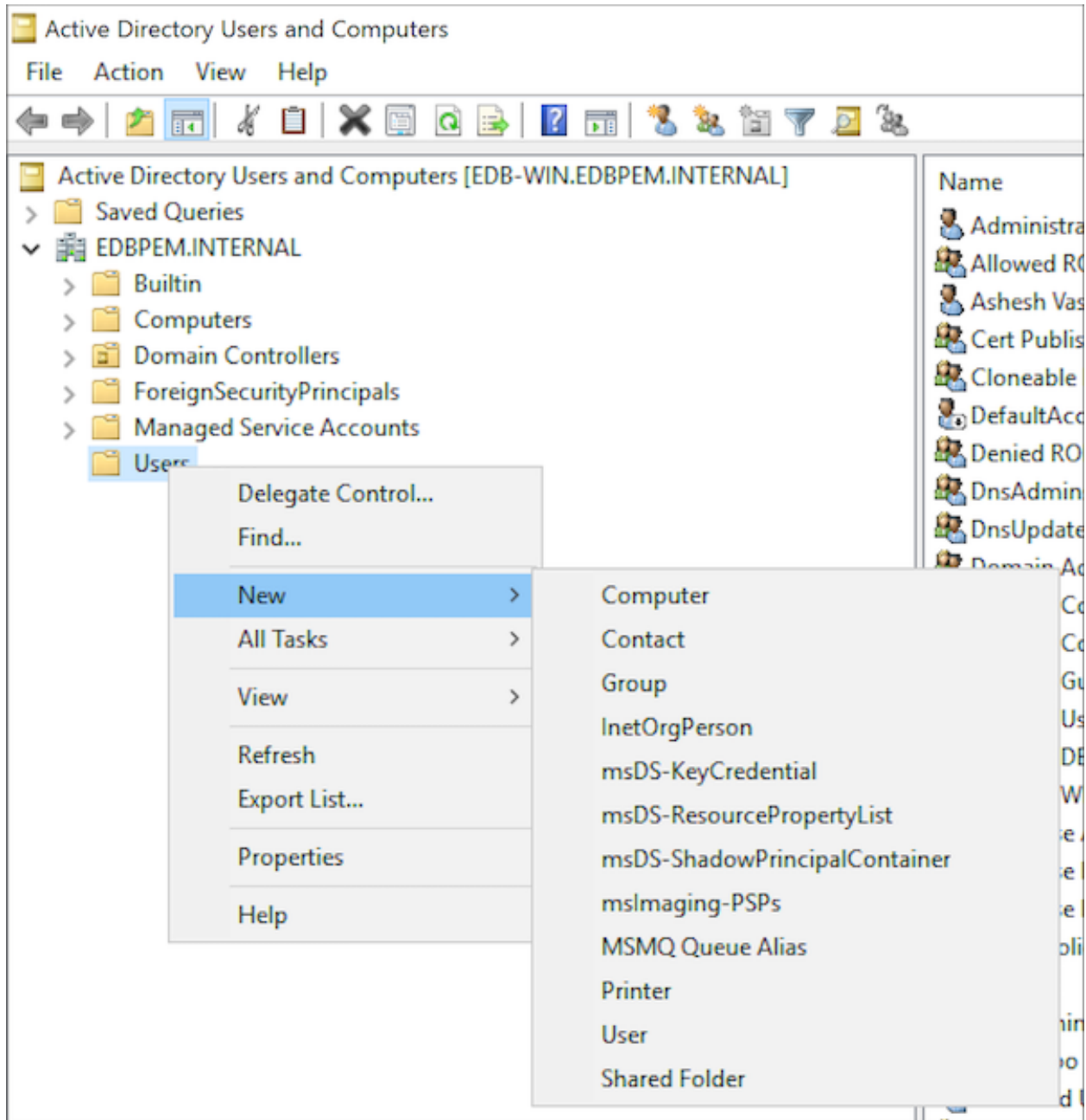


Fig. 3: Active Directory Users and Computers

3. Review the user details.
4. On the PEMServer Web Properties dialog box, add the users as members of the Domain Users group:

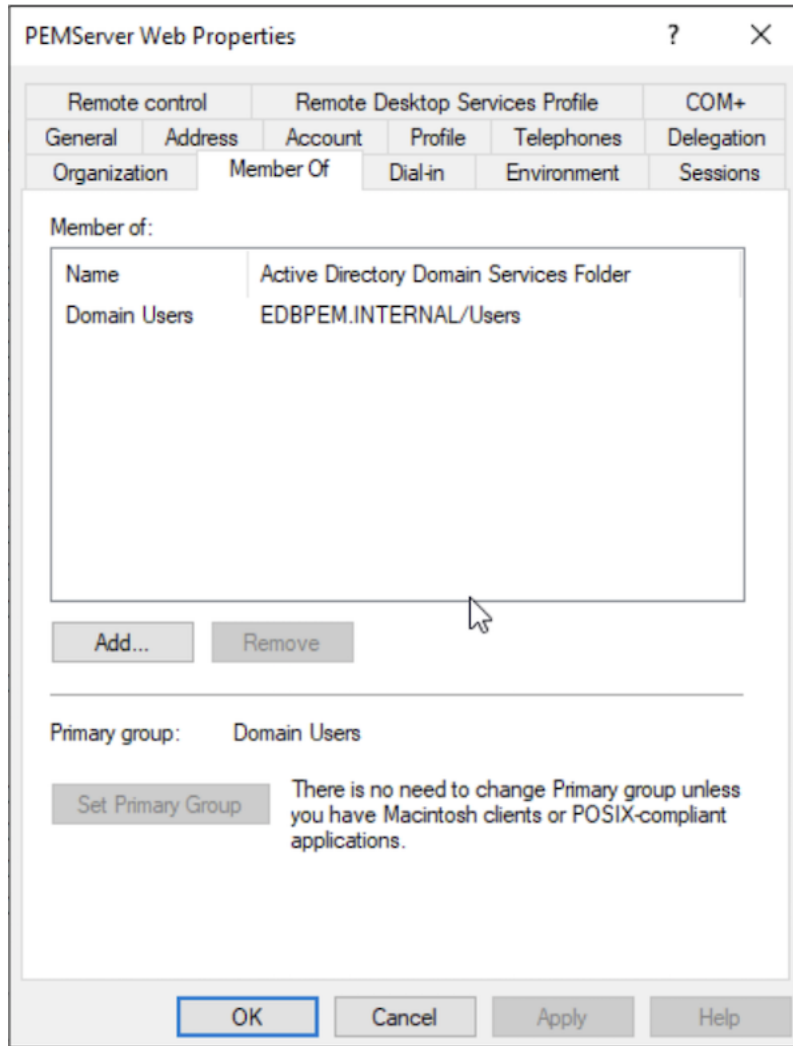


Fig. 4: PEM Server Web Properties

1. Create the user (for example, pemserverdb) in Active Directory of the Windows server to map with the Postgres service principal for the PEM backend database.

3. Extract key tables from Active Directory

Extract the key tables for the service principals and map them with the respective domain users you created.

1. Open Windows PowerShell as an administrator. Create a key table for HTTP service principal mapping with the user pemserverweb and a key table for Postgres service principal mapping with the user pemserverdb:

```
ktpass /out pemserver.keytab /princ HTTP/pem.edbpem.internal@EDBPEM.INTERNAL /mapuser_
↪ pemserverweb /crypto AES256-SHA1 +rndpass /target EDBPEM.INTERNAL -ptype KRB5_NT_
↪ PRINCIPAL -kvno 0
ktpass /out pemdb.keytab /princ postgres/pem.edbpem.internal@EDBPEM.INTERNAL /mapuser_
↪ pemserverdb /crypto AES256-SHA1 +rndpass /target EDBPEM.INTERNAL -ptype KRB5_NT_
↪ PRINCIPAL -kvno 0
```

(continues on next page)

(continued from previous page)

Where:

- ``pemserver.keytab`` is the name of the key table for the PEM web application. - ↵
- ↵ ``pemdb.keytab`` is the name of the key table for the PEM backend database server. - ↵
- ↵ ``pem.edbpeinternal@EDBPEINTERNAL`` is the hostname of the PEM server. Here, ↵
- ↵ `@EDBPEINTERNAL` means `@REALM`. - ``pemserverweb`` is the user for the PEM web ↵
- ↵ application. - ``pemserverdb`` is the user for the PEM backend database server. - ↵
- ↵ ``EDBPEINTERNAL`` is the domain of the target.

```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\windows\system32> ktpass /out pemserver.keytab /princ HTTP/pem.edbpeinternal@EDBPEINTERNAL /mapuser pemserverweb /crypto AES256-SHA1 +rndpass /target EDBPEINTERNAL -ptype KRB5_NT_PRINCIPAL
Using legacy password setting method
Successfully mapped HTTP/pem.edbpeinternal to pemserverweb.
Key created.
Output keytab to pemserver.keytab:
Keytab version: 0x502
keysize 91 HTTP/pem.edbpeinternal@EDBPEINTERNAL ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x12 (AES256-SHA1) keylength 32 (0x30b218d9e06e10d382e2b5859de1787e4bc56fe0bc797dd7beb29476aaf3e477)
PS C:\windows\system32>
  
```

Fig. 5: Windows PowerShell ktpass

Note: The command line argument `+rndpass` resets the password for the domain user `pemserverweb` to a random password. The `/target` option is optional.

1. On the **Accounts** tab, add Kerberos support for the user accounts.

Note: On the **Accounts** tab, the user logon name shows `HTTP/pem.edbpeinternal@EDBPEINTERNAL`. The **Delegation** tab is enabled for the `pemserverweb` user.

1. On the **Delegation** tab, select **Trustthisuserfordelegationtoanyservice(Kerberosonly)** for the users you created.
2. Copy both the key tables to the PEM server host or to the PEM web server and PEM backend database server hosts if installed on different hosts.
3. On the PEM server, move the key tables to the required location and change the ownership:

```

mv /tmp/pemserver.keytab <PEM_INSTALLATION_DIRECTORY>/share
chown pem <PEM_INSTALLATION_DIRECTORY>/share/pemserver.keytab
  
```

```

mv /tmp/pemdb.keytab <DATA_DIRECTORY_OF_POSTGRES>/
chown enterprisedb <DATA_DIRECTORY_OF_POSTGRES>/pemdb.keytab
  
```

Where:

- ``OS_USERNAME_ON_PEM_SERVER`` is the name of the operating system user on the PEM ↵
- ↵ server. - ``DATA_DIRECTORY_OF_POSTGRES`` is the path of the data directory of the ↵
- ↵ installed Postgres database (PostgreSQL/EDB Postgres Advanced Server).

PEMServer Web Properties

Organization	Member Of	Dial-in	Environment	Sessions
Remote control	Remote Desktop Services Profile			COM+
General	Address	Account	Profile	Telephones
		Delegation		

User logon name:
 @EDBPEM.INTERNAL

User logon name (pre-Windows 2000):

☐ Unlock account

Account options:

- ☐ Use only Kerberos DES encryption types for this account
- ☒ This account supports Kerberos AES 128 bit encryption.
- ☒ This account supports Kerberos AES 256 bit encryption.
- ☐ Do not require Kerberos preauthentication

Account expires

☒ Never

☐ End of:

Fig. 6: PEM Server Web Properties

4. Configure the PEM backend database server

Add the key table location in the `postgresql.conf` file.

```
krb_server_keyfile=FILE: /<DATA_DIRECTORY_OF_POSTGRES>/pemdb.keytab
```

Where `DATA_DIRECTORY_OF_POSTGRES` is the path of the data directory of the installed Postgres database (PostgreSQL/EDB Postgres Advanced Server).

Edit the `krb5.conf` file:

```
$ sudo vim /etc/krb5.conf
[libdefaults]
default_realm = EDBPEM.INTERNAL
Forwardable = True

[domain_realm]
.edbpem.org = EDBPEM.INTERNAL
edbpem.org = EDBPEM.INTERNAL

[realms]
EDBPEM.INTERNAL = {
    kdc = krb5server.edbpem.internal
    admin_server = krb5server.edbpem.internal
}
```

Restart the database server to reflect the changes:

```
systemctl restart <POSTGRES_SERVICE_NAME>
```

`POSTGRES_SERVICE_NAME` is the service name of the Postgres (PostgreSQL/EDB Postgres Advanced Server) database, for example, `postgresql-13` for PostgreSQL 13 database on CentOS or RHEL or Rocky Linux platforms.

5. Obtain and view the initial ticket

The `kinit` utility obtains and caches Kerberos tickets. You typically use this utility to obtain the ticket-granting ticket, using a password you entered to decrypt the credential from the key distribution center (KDC). The ticket-granting ticket is then stored in your credential cache.

You can view the details of the ticket using the `klist` utility.

Note: You must install the Kerberos client on the PEM server and the client machine before using `kinit` and `klist`.

```
$ kinit <USERNAME@REALM>
$ klist
```

It displays the principal along with the Kerberos ticket.

Note: The `USERNAME@REALM` specified here must be a database user having the `pem_admin` role and `CONNECT` privileges on the `pem` database.

6. Configure the PEM server

Run the PEM configure script on the PEM server to use Kerberos authentication:

```
$ sudo PEM_APP_HOST=pem.edbpem.internal PEM_KRB_KTNAME=<PEM_INSTALLATION_DIRECTORY>/share/
↪pemserver.keytab <PEM_INSTALLATION_DIRECTORY>/bin/configure-pem-server.sh
```

In the `config_setup.py` file, configure `PEM_DB_HOST` and check that the value of `PEM_AUTH_METHOD` is set to `'kerberos'`.

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/web/config_setup.py
PEM_DB_HOST=`pem.edbpem.internal`
```

Configure `HOST` in the `.install-config` file.

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/.install-config
HOST=`pem.edbpem.internal`
```

If the PEM server uses Kerberos authentication:

- All the monitored servers default to use the same authentication. To override the default, in the `config_local.py` file, add the parameter `ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS` and set it to `True`.
- All the authenticated user principals are appended with the realm (`USERNAME@REALM`) and passed as the database user name by default. To override the default, in the `config_local.py` file, add the parameter `PEM_USER_KRB_INCLUDE_REALM` and set it to `False`.

Restart the Apache server:

```
sudo systemctl restart <SERVICE_NAME>
```

Edit the entries at the top in `pg_hba.conf` to use the gss authentication method. Then reload the database server.

```
host    pem          +pem_user    <ip_of_pem_server>/32    gss
host    postgres     +pem_user    <ip_of_pem_server>/32    gss
```

```
systemctl reload <POSTGRES_SERVICE_NAME>
```

`POSTGRES_SERVICE_NAME` is the service name of the Postgres (PostgreSQL/EDB Postgres Advanced Server) database, for example, `postgres-13` for PostgreSQL 13 database on CentOS or RHEL or Rocky Linux platforms.

Note: You can't specify the connection type as `hostgssenc`. Windows doesn't support gss encrypted connection.

7. Browser settings

Configure the browser on the client machine to access the PEM web client to use the Spnego/Kerberos.

For Mozilla Firefox:

- Open the low-level Firefox configuration page by loading the `about:config` page.
- In the search box, enter `network.negotiate-auth.trusted-uris`.
- Double-click the `network.negotiate-auth.trusted-uris` preference and enter the hostname or the domain of the web server that's protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with commas.

- In the search box, enter `network.negotiate-auth.delegation-uris`.
- Double-click the `network.negotiate-auth.delegation-uris` preference and enter the hostname or the domain of the web server that's protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with commas.
- Select **OK**.

For Google Chrome on Linux or MacOS:

- Add the `--auth-server-whitelist` parameter to the `google-chrome` command. For example, to run Chrome from a Linux prompt, run the `google-chrome` command as follows:

```
google-chrome --auth-server-whitelist = "hostname/domain"
```

- After configuring the PEM server, you can access the PEM web interface in your browser. Navigate to:

```
https://<ip_address_of_PEM_server>:8443/pem
```

Note: You might see the following error while connecting to your Postgres cluster:

```
psql -h hostname template1      psql:  GSSAPI continuation error:  Unspecified GSS failure.
Minor code may provide more information      GSSAPI continuation error:  Key version is not
available
```

Add encryption types to the keytab using `ktutil` or by recreating the Postgres keytab with all crypto systems from AD.

3.3.2 Configuring the PEM server to use Kerberos authentication

You can configure Kerberos authentication for the PEM server. The Kerberos server works with hostnames and not with IP addresses. To use single sign-on in PEM server using Kerberos authentication, configure the following machines with hostnames using the DNS (realm).

```
- Kerberos server - PEM server (PEM web server and PEM backend database server) -
↔ Client machine
```

For example, if the realm on Kerberos server is `edbpem.org`, then you can set the Kerberos server hostname to `Krb5server.edbpem.org`, the PEM server hostname to `pem.edbpem.org`, and the client's hostname to `pg12.edbpem.org`. The convention is to use the DNS domain name as the name of the realm.

1. Install Kerberos, the PEM server, and the PEM backend database

Install Kerberos on the machine that functions as the authentication server. Install the PEM server on a separate machine. For more information, see [Installing the PEM Server](#).

Install the PEM backend database (Postgres/EDB Postgres Advanced Server) on the same machine as the PEM server or on a different one. For more information, see the [Installation steps on EDB Docs website](#).

2. Add principals on Kerberos server

Add the principals for the PEM web application deployed under an Apache web server (HTTPD/Apache2) and the PEM Backend Database Server (PostgreSQL/EDB Postgres Advanced Server).

```
$ sudo kadmin.local -q "addprinc -randkey HTTP/<HOSTNAME_OF_PEM_SERVER>"
$ sudo kadmin.local -q "addprinc -randkey postgres/<HOSTNAME_OF_PEM_SERVER>"
```

HOSTNAME_OF_PEM_SERVER must contain the realm of the Kerberos server. For example, you can specify pemdb.edbpem.org as the hostname of PEM server, with edbpem.org as the realm.

Note: If the PEM web application and the PEM backend database server are on different machines, then hostname is different for each one.

3. Extract key tables from Kerberos server

Extract the key tables from Kerberos for the PEM web application and the PEM backend database server:

```
sudo kadmin.local "ktadd -k <NAME_OF_PEM_WEB_FILE>.keytab HTTP/<HOSTNAME_OF_PEM_SERVER>"
sudo kadmin.local "ktadd -k <NAME_OF_PEM_DB_FILE>.keytab postgres/<HOSTNAME_OF_PEM_SERVER>"
```

Copy the key tables from the Kerberos server to the PEM server:

```
scp <NAME_OF_PEM_WEB_FILE>.keytab <OS_USERNAME_ON_PEM_SERVER>@<HOSTNAME_OF_PEM_SERVER>:/tmp
scp <NAME_OF_PEM_DB_FILE>.keytab <OS_USERNAME_ON_PEM_SERVER>@<HOSTNAME_OF_PEM_SERVER>:/tmp
```

On the PEM server, move the key tables to the required location and change ownership:

```
mv /tmp/<NAME_OF_PEM_WEB_FILE>.keytab <PEM_INSTALLATION_DIRECTORY>/share
chown pem <PEM_INSTALLATION_DIRECTORY>/share/<NAME_OF_PEM_WEB_FILE>.keytab
```

```
mv /tmp/<NAME_OF_PEM_DB_FILE>.keytab <DATA_DIRECTORY_OF_POSTGRES>/
chown enterprisedb <DATA_DIRECTORY_OF_POSTGRES>/<NAME_OF_PEM_DB_FILE>.keytab
```

Where:

- NAME_OF_PEM_WEB_FILE is the name specified for the key table for the PEM web application.
- NAME_OF_PEM_DB_FILE is the name specified for the key table for the PEM backend database server.
- OS_USERNAME_ON_PEM_SERVER is the name of the operating system user on the PEM server.
- DATA_DIRECTORY_OF_POSTGRES is the path of the data directory of the installed Postgres database (PostgreSQL/EDB Postgres Advanced Server).

4. Configure the PEM backend database server

Add the key table location in the `postgresql.conf` file:

```
krb_server_keyfile=FILE: /<DATA_DIRECTORY_OF_POSTGRES>/<NAME_OF_PEM_DB_FILE>.keytab
```

Where:

- `NAME_OF_PEM_DB_FILE` is the name specified for the key table for the PEM backend database server.
- `DATA_DIRECTORY_OF_POSTGRES` is the path of the data directory of the installed Postgres database (PostgreSQL/EDB Postgres Advanced Server).

Edit the `krb5.conf` file:

```
$ sudo vim /etc/krb5.conf
[libdefaults]
    default_realm = EDBPEM.ORG
    Forwardable = True

[domain_realm]
    .edbpem.org = EDBPEM.ORG
    edbpem.org = EDBPEM.ORG

[realms]
    EDBPEM.ORG = {
        kdc = krb5server.edbpem.org
        admin_server = krb5server.edbpem.org
    }
```

Restart the database server to reflect the changes:

```
systemctl restart <POSTGRES_SERVICE_NAME>
```

`POSTGRES_SERVICE_NAME` is the service name of the Postgres (PostgreSQL/EDB Postgres Advanced Server) database, for example, `postgresql-13` for PostgreSQL 13 database on CentOS or RHEL or Rocky Linux platforms.

5. Obtain and view the initial ticket

The `kinit` utility obtains and caches Kerberos tickets. You typically use this utility to obtain the ticket-granting ticket, entering a password to decrypt the credential from the key distribution center (KDC). The ticket-granting ticket is then stored in your credential cache.

You can view the details of the ticket using the `klist` utility.

Note: Install the Kerberos client on the PEM server and the client machine before using `kinit` and `klist`.

```
$ kinit <USERNAME@REALM>
$ klist
```

It displays the principal along with the Kerberos ticket.

Note: The `USERNAME@REALM` specified here must be a database user having the `pem_admin` role and `CONNECT` privilege on `pem` database.

6. Configure the PEM server

Run the PEM configure script on the PEM server to use Kerberos authentication:

```
$ sudo PEM_APP_HOST=<HOSTNAME_OF_PEM_SERVER> PEM_KRB_KTNAME=<PEM_INSTALLATION_DIRECTORY/  
→share/<NAME_OF_PEM_WEB_FILE>.keytab <PEM_INSTALLATION_DIRECTORY>/bin/configure-pem-  
→server.sh
```

Configure `PEM_DB_HOST` in the `config_setup.py` file. Check that the value of `PEM_AUTH_METHOD` is set to `'kerberos'`.

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/web/config_setup.py  
PEM_DB_HOST=`<HOSTNAME_OF_PEM_SERVER>`
```

Configure the host in the `.install-config` file:

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/.install-config  
HOST=`<HOSTNAME_OF_PEM_SERVER>`
```

If the PEM server uses Kerberos authentication:

- All the monitored servers default to use the same authentication. To override the default, in the `config_local.py` file, add the parameter `ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS` and set it to `True`.
- All the authenticated user principals are appended with the realm (`USERNAME@REALM`) and passed as the database user name by default. To override the default, in the `config_local.py` file, add the parameter `PEM_USER_KRB_INCLUDE_REALM` and set it to `False`.
- Restart the Apache server

```
sudo systemctl restart <SERVICE_NAME>
```

- Edit the entries at the top of `pg_hba.conf` to use the gss authentication method, and reload the database server.

```
host pem      +pem_user    <ip_of_pem_server>/32  gss  
host postgres +pem_user    <ip_of_pem_server>/32  gss
```

```
systemctl reload <POSTGRES_SERVICE_NAME>
```

`POSTGRES_SERVICE_NAME` is the service name of the Postgres (PostgreSQL/EDB Postgres Advanced Server) database, for example, `postgresql-13` for PostgreSQL 13 database on CentOS or RHEL or Rocky Linux platforms.

Note: If you're using PostgreSQL or EDB Postgres Advanced Server 12 or later, then you can specify connection type as `hostgssenc` to allow only gss-encrypted connection.

7. Browser settings

Configure the browser on the client machine to access the PEM web client to use the Spnego/Kerberos.

For Mozilla Firefox:

1. Open the low-level Firefox configuration page by loading the `about:config` page.
2. In the search box, enter `network.negotiate-auth.trusted-uris`.
3. Double-click the `network.negotiate-auth.trusted-uris` preference and enter the hostname or the domain of the web server that's protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with a comma.
4. In the search box, enter `network.negotiate-auth.delegation-uris`.
5. Double-click the `network.negotiate-auth.delegation-uris` preference and enter the hostname or the domain of the web server that's protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with a comma.
6. Select **OK**.

For Google Chrome on Linux or MacOS:

- Add the `--auth-server-whitelist` parameter to the `google-chrome` command. For example, to run Chrome from a Linux prompt, run the `google-chrome` command as follows:

```
google-chrome --auth-server-whitelist = "hostname/domain"
```

- After configuring the PEM server, you can access the PEM web interface in your browser. Navigate to:

```
https://<ip_address_of_PEM_server>:8443/pem
```

Note: You might see the following error while connecting to your Postgres cluster:

```
psql -h hostname template1      psql:  GSSAPI continuation error:  Unspecified GSS failure.
Minor code may provide more information      GSSAPI continuation error:  Key version is not
available
```

Add encryption types to the keytab using `ktutil` or by recreating the Postgres keytab with all crypto systems from AD.

3.3.3 Configuring the PEM server to use two-factor authentication

PEM supports two methods for 2FA:

- Email authentication
- Authenticator app (such as Google Authenticator)

To enable 2FA, you can copy these settings from the `config.py` file to the `config_local.py` file and modify the following parameters.

Parameter	Description
<code>MFA_ENABLED</code>	Set to <i>true</i> to enable the two-factor authentication. Default value is <i>false</i> .
<code>MFA_FORCE_REGISTRATION</code>	Set to <i>true</i> to ask the users to register forcefully for the two-factor authentication methods at login. Default value is <i>false</i> .

continues on next page

Table 2 – continued from previous page

Parameter	Description
MFA_SUPPORTED_METHODS	Set to <i>email</i> to use the email authentication method (send a one-time code by email) or <i>authenticator</i> to use the TOTP-based application authentication method.
MFA_EMAIL_SUBJECT	Set to the subject of the email for email authentication. Default value is <i><APP_NAME> - Verification Code</i> .

Mail server configuration

To use the email authentication method, you need to configure mail server settings.

PEM server can send an email using either the SMTP configurations saved in the PEM configuration or using Flask-Mail.

To send the email verification code using the internal SMTP configuration from the PEM configuration, set the parameter `MAIL_USE_PEM_INTERNAL` to `True`. If set to `False`, the following mail configuration is used to send the code on the user-specified email address:

- `MAIL_SERVER` = 'localhost'
- `MAIL_PORT` = 25
- `MAIL_USE_TLS` = False
- `MAIL_USE_SSL` = False
- `MAIL_USERNAME` = None
- `MAIL_PASSWORD` = None
- `MAIL_DEFAULT_SENDER` = None

For more details about these configurations, see the [Flask-Mail documentation](#).

Note: PEM SMTP alerts don't use this configuration.

3.4 Securing your deployment

When you configure PEM, there are security practices to consider. These security practices are optional. However, EDB recommends these practices to enhance the overall system security.

PEM depends on third-party components from the vendor repository, including the Apache web server, OpenSSL, snmp++, and libcurl. To ensure these components are up to date, update your operating system and regularly apply security updates to avoid any security vulnerability. Without the most recent security patches, your system is vulnerable to cyberattacks. Security patches protect your devices and their data by applying the latest updates to guard against the latest threats.

Some of the benefits of regularly applying security patches include:

- Reducing exposure to cyberattacks
- Avoiding lost productivity
- Protecting data from malware like ransomware
- Avoiding worm infections that use security loopholes to spread over the network

3.4.1 Apache HTTPD security configurations

On Windows, Apache HTTPD is named PEM HTTPD. The Apache HTTPD configuration file is `pme.conf` and the SSL configuration file is `httpd-ssl-pem.conf`. Both configuration files are in the `<Apache_Installation_Path>/conf/addons` directory.

On Linux, the Apache HTTPD configuration file is `edb-pem.conf` and the SSL configuration file is `edb-ssl-pem.conf`. Both configurations files are in the `<Apache_Installation_Path>/conf.d` directory.

Disable SSLv2 and SSLv3

Disable SSL versions SSLv2, SSLv3, TLS 1, and TLS 1.1. These versions are the most vulnerable and have cryptographic concerns.

To disable the versions, add the following command to the Apache HTTPD configuration file:

```
SSLProtocol -ALL +TLSv1.2
```

Restart the web server to apply the changes to the configuration file.

PEM adds the following lines to the SSL configuration file to allow for the use of TLS 1.2 as security:

```
SSLProtocol -All TLSv1.2
```

```
SSLProxyProtocol -All TLSv1.2
```

Secure HTTPD with SSL certificates

We recommend an additional layer of SSL security for the web application.

During PEM installation, PEM generates and uses self-signed certificates for the Apache/HTTPD server. If you want to use your own SSL certificate for PEM, you must update the Apache configuration file.

On Linux, you need to open the Apache HTTPD configuration file (`edb-ssl-pem.conf`) in a text editor and be a user with write permission on the file. You must also change the server name and file names in the configuration file to match your certificate files.

Update these two SSL directives in the PEM VirtualHost section:

- `SSLCertificateFile` is your DigiCert certificate file (for example, `your_domain_name.crt`).
- `SSLCertificateKeyFile` is the `.key` file generated when you created the CSR (for example, `your_private.key`).

For example, make the following updates:

```
SSLEngine on
SSLCertificateFile /path/to/your_domain_name.crt
SSLCertificateKeyFile /path/to/your_private.key
```

You can also replace the HTTPD self-signed SSL certificates with trusted CA-signed certificates in PEM. For instructions, see this [Postgres tutorial](#).

Disable web server information exposure

We recommend that you disable all web server signatures as part of web server security. To disable the web server signature, add the following parameters to the Apache HTTPD configuration file. PEM disables exposure of the information by adding the parameters to the Apache HTTPD configuration file.

```
ServerTokens Prod
ServerSignature Off
```

The `ServerTokens` directive controls the server response header field, which returns to the client. We recommend that you hide the Apache server version by adding this parameter in the Apache HTTPD configuration file.

The `ServerSignature` directive includes a footer for server-produced documents. The footer contains information regarding the Apache configuration, like the Apache and operating system version. To limit the exposure of this information, we recommend that you disable the directive in the Apache HTTPD configuration file.

Restart the web server to apply any changes to the Apache HTTPD configuration file.

Disable directory listing

The directory listing allows an attacker to view the complete contents of the directory. By default, the web server enables this listing, allowing an attacker to discover and view any file. This listing might lead to the attacker reverse engineering an application to obtain the source code, analyze it for possible security flaws, and discover more information about an application.

To avoid this risk, disable the directory listing by setting the `Options` directive in the Apache HTTPD configuration file. PEM disables the directory listing by setting the following option in the web server configuration file:

```
<Directory /application/directory> Options -Indexes </Directory>
```

Restart the web server to apply the changes made to the configuration file.

Restrict the access to a network or IP address

Apache allows you to provide the client hostnames or IP addresses that can access the application. To give access to an IP address or network, enter the network address in the `Allow` directive.

```
<Directory /application/hostname>
Options None
AllowOverride None
Order deny,allow
Deny from all
Allow from 192.168.0.0/24
</Directory>
```

The `ALLOWED_HOSTS` configuration parameter in the application configuration file sets the allowed hosts by IP address. The application configuration file (`config_local.py`) is located in `<PEM_INSTALLATION_PATH>/web`.

PEM allows all the hosts to connect with the application.

For example:

```
# You can set the range of IP addresses in the configuration file:

ALLOWED_HOSTS = [225.0.0.0/8, 226.0.0.0/7, 228.0.0.0/6]
```

(continues on next page)

(continued from previous page)

```
# You can set the IP addresses to allow a host on a subnet level in the configuration_
↪file:
ALLOWED_HOSTS = [192.0.2.0/28, ::192.0.2.0/124]

# You can set a specific individual host address (based on the IP address) in the_
↪configuration file:
ALLOWED_HOSTS = [127.0.0.1, 192.168.0.1]
```

To apply the application configuration file changes, restart the web server.

Cross-site tracing

You can use two HTTP methods to debug the web server connections: TRACE and TRACK. When an HTTP TRACE request is sent to a supported web server, the server responds and echoes the data passed to it, including any HTTP headers. We recommend that you disable these methods in the Apache configuration.

To disable the TRACE method for all virtual hosts, add the following line to the Apache HTTPD configuration file:

```
TraceEnable off
```

To disable these methods for a specific virtual host, add the following lines for each virtual host in the Apache configuration file. PEM also adds the following lines to the Apache HTTPD configuration file:

```
RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK|OPTIONS)
RewriteRule .* - [F]
```

Run web server from a nonprivileged user account

Running the Apache web service as a root user can create a security issue. We recommend that you run the web server as a unique nonprivileged user. Doing so helps to secure any other services running during a security breach.

PEM runs as a WSGI application. To delegate the WSGI applications that are running, create distinct daemon processes using the `WSGIDaemonProcess` directive.

On Linux, the Apache web server starts as the root user. Meanwhile, PEM runs the daemon processes as the pem user. On Windows, the `WSGIDaemonProcess` directive and features aren't available. During the installation, PEM HTTPD installs as a service. For this service installation to succeed, you must be a member of the Administrators group.

By default, the Apache services run as the system user (the `LocalSystem` account).

Customize security HTTP headers in PEM web server

PEM contains a separate configuration file to fix the following security issues. We recommend that you override the configuration only of `config_local.py` and not of `config.py`. The `config_local.py` file is present only on some systems. You must create it to override the application-level configurations. During a PEM upgrade, you can't overwrite `config_local.py`, but you can overwrite changes in `config.py` and `config_distro.py`. Remove `config_local.py` after uninstalling PEM.

By default, `config_local.py` is in `/usr/edb/pem/web` on Linux and at `C:\ProgramFiles\edb\pem\server\share\web` on Windows.

Host header injection attacks

HTTP host header attacks exploit vulnerable websites that handle the host header value in an unsafe way. If the server implicitly trusts the host header and fails to validate or escape it properly, an attacker can use this input to inject harmful payloads that manipulate server-side behavior. The web applications typically don't know the domain they are deployed on unless specified in a configuration file during setup. When they need to know the current domain, for example, they can resort to retrieving the domain from the host header to generate an absolute URL. The host header is a potential vector for exploiting a range of other vulnerabilities, most notably web cache poisoning and SQL injections.

X-Frame-Options

X-Frame-Options indicate whether a browser is allowed to render a page in an `<iframe>` tag. It specifically protects against clickjacking. PEM has a host validation `X_FRAME_OPTIONS` option to prevent these attacks, which you can configure in the `config_local.py` file. The default is:

```
X_FRAME_OPTIONS = "SAMEORIGIN"
```

Content-Security-Policy

Content-Security-Policy is part of the HTML5 standard. It provides a broader range of protection than the X-Frame-Options header, which it replaces. It is designed so that website authors can whitelist domains. The authors can load resources (like scripts, stylesheets, and fonts) from the whitelisted domains and also from domains that can embed a page.

PEM has a host validation `CONTENT_SECURITY_POLICY` option to prevent attacks, which you can configure in the `config_local.py` file. The default is:

```
CONTENT_SECURITY_POLICY = "default-src https: data: blob: unsafe-inline 'unsafe-eval;'"
```

Strict-Transport-Security

The Strict-Transport-Security (HSTS) response header can prevent a man-in-the-middle attack. When you enable the option, websites or web applications tell browsers that they accept only HTTPS and not HTTP. The default is:

```
STRICT_TRANSPORT_SECURITY = "max-age=31536000;includeSubDomains"
```

Note: Adding this parameter can cause problems if config is changed. Therefore, we recommend that you add it only after PEM installation is complete and tested.

X-Content-Type-Options

The X-Content-Type-Options response HTTP header is a marker. The server uses the marker to indicate that the MIME types advertised in Content-Type headers can't be changed and followed. The following is a way to opt out of MIME type sniffing, that is, to say that the MIME types are deliberately configured. The default is:

```
X_CONTENT_TYPE_OPTIONS = "nosniff"
```

X-XSS-Protection

Cross-site scripting (XSS) is one of the most common application layer vulnerabilities in the web servers. XSS enables attackers to inject client-side scripts into web pages that other users view. The HTTP X-XSS-Protection response to the header is a feature of Internet Explorer, Chrome, and Safari. It stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. These protections are unnecessary in modern browsers when sites implement a strong Content-Security-Policy that disables the use of inline JavaScript ('unsafe-inline'). However, these protections can still provide protections for users of older web browsers that don't yet support CSP. The default is:

```
X_XSS_PROTECTION = "1; mode=block"
```

To avoid this, add the following options to the Apache configuration file:

```
<IfModule mod_headers.c>
Header set X-XSS-Protection "1; mode=block"
</IfModule>
```

To apply the configuration file changes, restart the web server.

By default, PEM sets X-XSS-Protection to "1; mode=block" in the application configuration file, which is located at /usr/edb/pem/web/config.py.

To apply the changes, restart the Apache service.

For detailed information on the config.py file, see Managing configuration settings.

Cookie security

Cookies are small packets of data that a server sends to your browser to store configuration data. The browser sends them and all other requests to the same server, so it's important to know how to secure cookies. Multiple configuration options in config.py can make cookies secure. These are the three most important options:

- **SESSION_COOKIE_SECURE** — The flag prevents cookies from sending over an unencrypted connection. The browser can't add the cookie to any request to a server without an encrypted channel. The browser can add cookies only to connections such as HTTPS. The default is:

```
SESSION_COOKIE_SECURE = True
```

- **SESSION_COOKIE_HTTPONLY** — By default, JavaScript can read the content of cookies. The HTTPOnly flag prevents scripts from reading the cookie. Instead, the browser uses the cookie only with HTTP or HTTPS requests. Hackers can't exploit XSS vulnerabilities to learn the contents of the cookie. For example, the sessionId cookie never requires that it be read with a client-side script. So, you can set the HTTPOnly flag for sessionId cookies. The default is:

```
SESSION_COOKIE_HTTPONLY = True
```

- **ENHANCED_COOKIE_PROTECTION** — When you set this option to **True**, then a token is generated according to the IP address and user agent. In all subsequent requests, the token recalculates and compares to the one computed for the first request. If the session cookie is stolen and the attacker uses it from another location, the generated token is different. In that case, the extension clears the session and blocks the request. The default is:

`ENHANCED_COOKIE_PROTECTION = True`

Note: This option can cause problems when the server deploys in dynamic IP address hosting environments, such as Kubernetes or behind load balancers. In such cases, set this option to *False*.

To apply the changes, restart the Apache service.

For detailed information on `config.py` file, see [Managing Configuration Settings](#).

3.4.2 PEM application security configurations

Session timeout

Insufficient session expiration by the web application increases the exposure of other session-based attacks. The attacker has more time to reuse a valid session ID and hijack the associated session. The shorter the session interval is, the less time an attacker has to use the valid session ID. We recommend that you set the inactivity timeout for the web application to a low value to avoid this security issue.

In PEM, you can set the timeout value for a user session. When there's no user activity for a specified duration on the web console, PEM logs out the user from the web console. A PEM administrator can set the length of time for inactivity. This value is for the whole application and not for each user. To configure the timeout duration, modify the `USER_INACTIVITY_TIMEOUT` parameter in the `config_local.py` file, located in the `<PEM_INSTALLATION_PATH>/web` directory. By default, this functionality is disabled.

For example, to specify for an application to log out a user after 15 minutes of inactivity, set:

`USER_INACTIVITY_TIMEOUT = 900`

Note: The timeout value is specified in seconds.

To apply the changes, restart the Apache service.

For detailed information on the `config.py` file, see [Managing Configuration Settings](#).

RestAPI header customization

You can customize the RestAPI token headers to meet your requirements. The default values aren't exposed by the `config.py` file. Customize the following headers in the `config_local.py` file:

PEM_HEADER_SUBJECT_TOKEN_KEY

This configuration option allows you to change the HTTP header name to get the generated token. By default, when you send a request to create a token, the server response has an `X-Subject-Token` header. This header contains the value of a newly generated token. If you want to customize the header name, then you can update the `config_local.py` file:

```
PEM_HEADER_SUBJECT_TOKEN_KEY = Pem-RestAPI-Generate-Token
```

This command produces the following output:

```
curl -ik -X POST -d {"username":"enterprisedb","password":"edb"} -H "Content-Type:↵
↵application/json" https://localhost:8443/pem/api/token/
HTTP/1.1 201 CREATED
Date: Thu, 29 Oct 2020 11:03:48 GMT
Server: Apache
Content-Length: 326
Pem-RestAPI-Generate-Token: 997aef95-d46d-4d84-932a-a80146eaf84f
```

PEM_HEADER_TOKEN_KEY

This configuration option allows you to change the HTTP request header name. With this header name, you can send the token to the PEM server. By default, when you send a request to generate a token, the token header name is `X-Auth-Token`. If you want to customize the RestAPI request header name, then you can update the `config_local.py` file:

```
PEM_HEADER_TOKEN_KEY = Pem-Token
```

This setting allows you to send the token:

```
$ curl -Lk -X GET -H "Pem-Token: gw5rzaloxyp91tttd1c97w24b5sv60cllc24sxy9" https://
↵localhost:8443/pem/api/v4/agent
```

PEM_TOKEN_EXPIRY

This configuration option allows you to change the PEM RestAPI token expiry time after it's generated. By default, the token expiry time is set to 20 minutes (1200 seconds). If you want to change the token expiry time to 10 minutes, then you can update the `config_local.py` file:

```
PEM_TOKEN_EXPIRY = 600
```

To apply the changes, restart the Apache service.

Role-based access control in PEM

Role-based access control (RBAC) restricts application access based on a user's role in an organization and is one of the primary methods for access control. The roles in RBAC refer to the levels of access that users have to the application. Users are allowed to access only the information needed to do their jobs. Roles in PEM are inheritable and additive, rather than subtractive. In other words, as a PEM admin you need to grant the lowest level role to the user and then grant the roles the user needs to perform their job. For example, to give access only to SQL profiler:

```
CREATE ROLE user_sql_profiler WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT,
↪ NOREPLICATION CONNECTION LIMIT -1 PASSWORD xxxxxx;
GRANT pem_user, pem_comp_sqlprofiler TO user_sql_profiler;
```

For detailed information on roles, see PEM Roles .

SQL/Protect plugin

Often, preventing an SQL injection attack is the responsibility of the application developer, while the database administrator has little or no control over the potential threat. The difficulty for database administrators is that the application must have access to the data to function properly.

SQL/Protect is a module that allows a database administrator to protect a database from SQL injection attacks. SQL/Protect examines incoming queries for typical SQL injection profiles in addition to the standard database security policies.

Attackers can perpetrate SQL injection attacks with several different techniques. A specific signature characterizes each technique. SQL/Protect examines queries for unauthorized relations, utility commands, SQL tautology, and unbounded DML statements. SQL/Protect gives the control back to the database administrator by alerting the administrator to potentially dangerous queries and then blocking those queries.

Note: This plugin works only on the EDB Postgres Advanced Server server, so this is useful only when your PEM database is hosted on the EDB Postgres Advanced Server server.

For detailed information about the SQL Profiler plugin, see
SQL Profiler plugin .

Password management

One security tip for PEM administrative users is to change your PEM login passwords to something new regularly. Changing your password:

- Prevents breaches of multiple accounts
- Prevents constant access
- Prevents the use of saved passwords on a physically unsecured system
- Limits access gained by keystroke loggers

Run pemAgent jobs with a non-root user

In most cases, pemAgent is installed as a root user and runs as a daemon process with root privileges. By default, PEM disables running the scheduled jobs/task. PEM provides support for running scheduled jobs as a non-root user by changing the pemAgent configuration file.

To run scheduled jobs as a non-root user, modify the entry for the `batch_script_user` parameter in the `agent.cfg` file and specify the user to run the script. You can either specify a non-root user or root user identity. If you don't specify a user, or the specified user doesn't exist, then the script doesn't execute. Restart the agent after modifying the file. If a non-root user is running pemagent, then the value of `batch_script_user` is ignored, and the same non-root user used for running the pemagent executes the script.

To invoke a script on a Windows system, set the registry entry for `AllowBatchJobSteps` to `true` and restart the PEM agent. PEM registry entries are located in:

```
HKEY_LOCAL_MACHINE\Software\Wow6432Node\EnterpriseDB\PEM\agent\
```

Changing the pemAgent and PEM backend database server certificates

By default, when you install PEM, the installer generates and uses self-signed certificates for the pemAgent and PEM database server. PemAgent uses these certificates when connecting to the PEM database server. To use your own SSL certificate for the pemAgent and PEM database server, see Managing certificates .

Note: PEM doesn't support placing the SSL CA certificates at a custom location. Don't change the location of `ca_certificate.crt` and `ca_key.key` .

3.5 Web server installation options

During the PEM server installation, you can specify your hosting preferences for the Apache web server.

3.5.1 PEM server and Apache web server on separate hosts

1. Install the PEM server on both the hosts. See Installing the PEM server on Linux .
2. Configure the PEM server host by selecting the **Database** option on the first host.
3. Configure an Apache web server by selecting the **WebServices** option on the second host.

For more information about configuring a PEM server, see Configuring the PEM server on Linux platforms .

Note: For production environments, best practice is to have the PEM server and Apache web server on separate hosts.
 ## PEM server and Apache web server on the same host

1. Install the PEM server. See Installing the PEM server on Linux .
2. Run the configuration script. Select the **WebServicesandDatabase** option to install the PEM server and Apache web server on the same host. See Configuring the PEM server on Linux platforms .

Note: For nonproduction environments, best practice is to have the PEM server and Apache web server on same hosts.
