



**Postgres Enterprise Manager -
Planning Guide**
Release 8.5.0

EDB

Jul 15, 2022

CONTENTS

1	Understanding Postgres Enterprise Manager components and architecture	3
1.1	PEM server	4
1.2	PEM agent	6
1.3	PEM web client	7
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
2.1.2	PEM server and Apache web server on the same host	10
2.1.2.1	Using Failover Manager for high availability	10
2.1.2.1.1	Initial product installation and configuration	10
2.1.2.1.2	Set up the primary node for streaming replication	11
2.1.2.1.3	Set up the standby nodes for streaming replication	12
2.1.2.1.4	Set up EFM to manage failover on all hosts	14
2.1.2.1.5	Current limitations	18
2.1.2.2	Connection pooling using pgBouncer	19
2.1.2.2.1	PEM server and agent connection management mechanism	19
2.1.2.2.2	Preparing the PEM database server	20
2.1.2.2.3	Configuring PgBouncer	21
2.1.2.2.4	Configuring the PEM agent	24
2.1.2.3	Authentication options	25
2.1.2.3.1	Configuring the PEM server to use Kerberos authentication	26
2.1.2.3.2	Configuring the PEM server to use Windows Active Directory Domain Services for Kerberos authentication (SSPI)	30
2.1.2.3.3	Configuring the PEM server to use two-factor authentication	42
2.1.2.4	Securing your deployment	43
2.1.2.4.1	Apache HTTPD security configurations	43
2.1.2.4.2	PEM application security configurations	49
2.1.2.5	Web server installation options	52
2.1.2.5.1	PEM server and Apache web server on separate hosts	52

Welcome to Postgres Enterprise Manager (PEM) Planning Guide. This guide describes architecture and considerations when you are planning to use PEM. 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) is a tool designed to monitor and manage multiple Postgres servers through a single GUI interface. PEM is capable of monitoring 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 the schema objects (tables, indexes, etc.) within them.

Note:

- The term Postgres refers to either PostgreSQL or EDB Postgres Advanced Server or EDB Postgres Extended Server.

PEM consists of a number of individual software components; the individual components are described below.

- ****PEMServer**** - The PEM Server is used as the data repository for monitoring data and as a server to which both Agents and Clients connect. The PEM server consists of an instance of PostgreSQL and an associated database for storage of 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 utilize PEM extended functionality. The web interface software is installed with the PEM server and is accessed via any supported web browser.
- ****SQLProfiler**** - SQL Profiler is a Postgres server plugin to record the monitoring data and query plans to be analyzed by the SQL Profiler tool in PEM. This is an optional component of PEM, but the plugin must be installed into each instance of Postgres with which you wish to use the SQL Profiler tool. The SQL Profiler may be used with any supported version of an EDB distribution of PostgreSQL server or Advanced Server (not just those managed through the PEM server). See the `ref:SQL Profiler Configuration </pem/latest/profiling-workloads/pem_sqlprofiler/>` for details and supported versions.

(continued from previous page)

PEM Architecture

The following architectural diagram illustrates the relationships between the PEM server, clients, and managed as well as unmanaged Postgres servers.

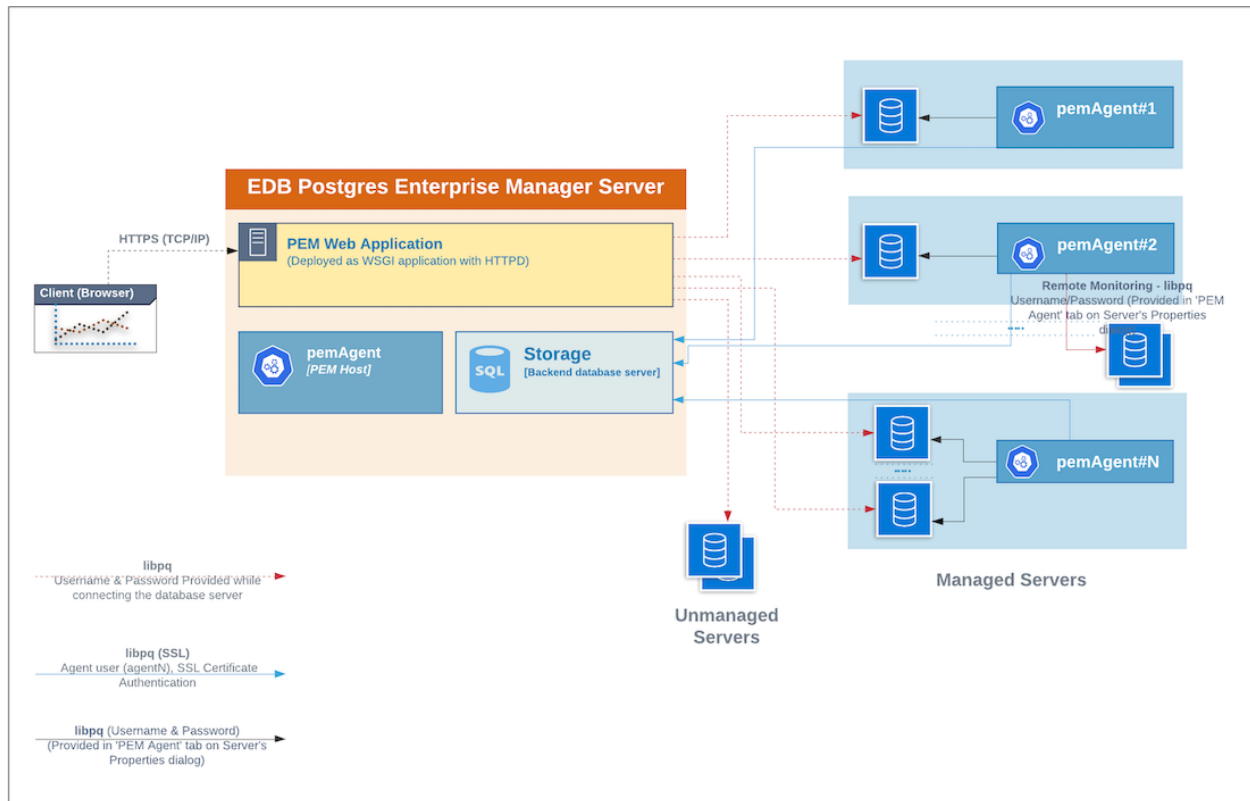


Fig. 1: PEM Architecture

1.1 PEM server

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 utilizes 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.

- **Postgres Instance (Database server)** - 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. .. Note:

All the PEM features are available irrespective of which backend database server you select, PostgreSQL or EDB Postgres Advanced Server.

- ****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.

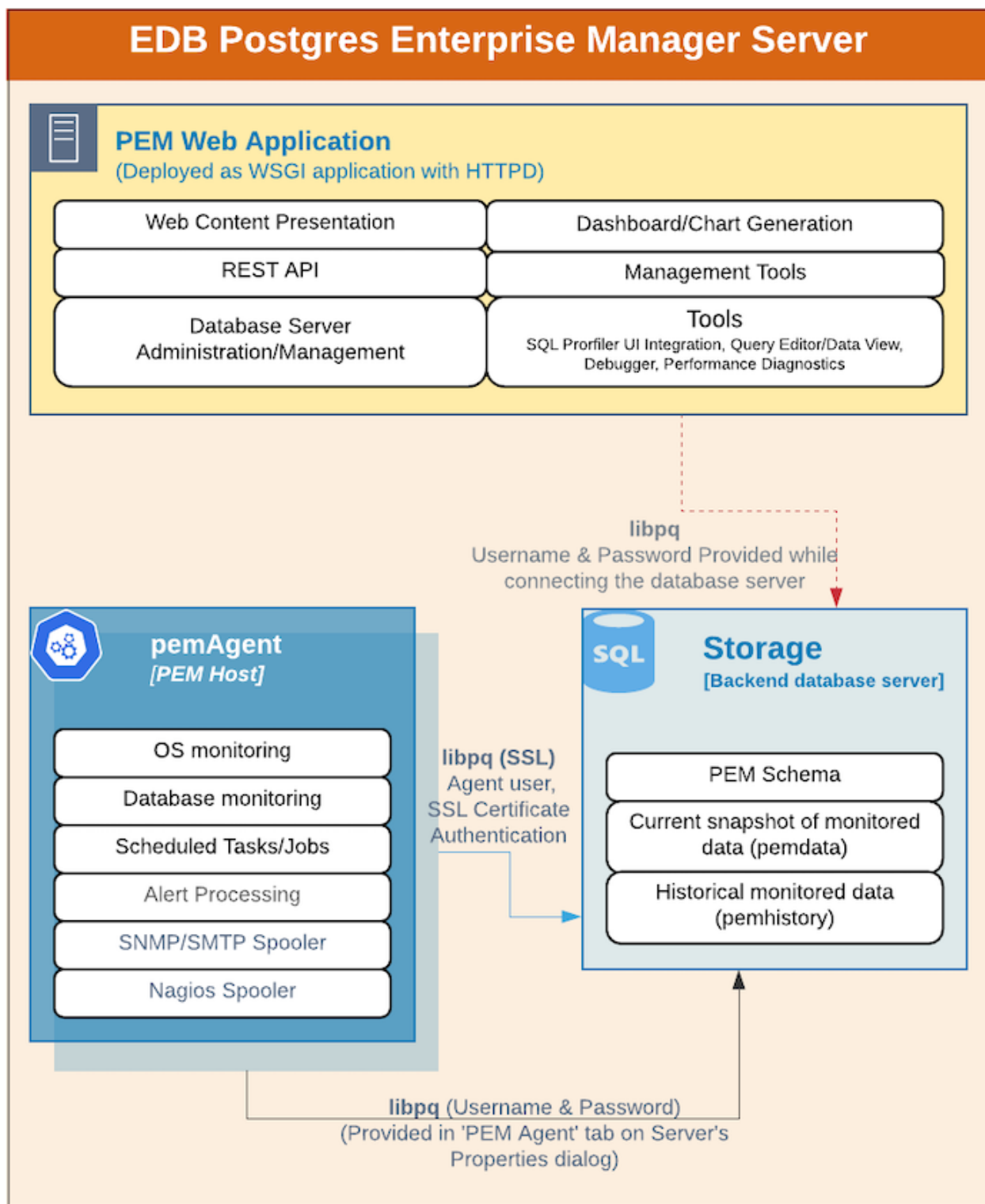


Fig. 2: PEM Server

(continued from previous page)

- **ApacheWebServer(HTTDPD)** - The PEM Web Application **is** deployed as a WSGI application **with** HTTPD to provide web services to the client. It **is** comprised of the following:
 - **Webcontentpresentation** - The presentation layer **is** created by the Web Application (**for** example Browser, login page,..).
 - **RestAPI** - The REST API allows integration **with** other apps **and** services.
 - **DatabaseServerAdministration/Management** - Database server administration **and** management activities like CREATE, ALTER, DROP, etc. can be performed **for** managed **as well as** 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 made 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 may 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.

1.2 PEM agent

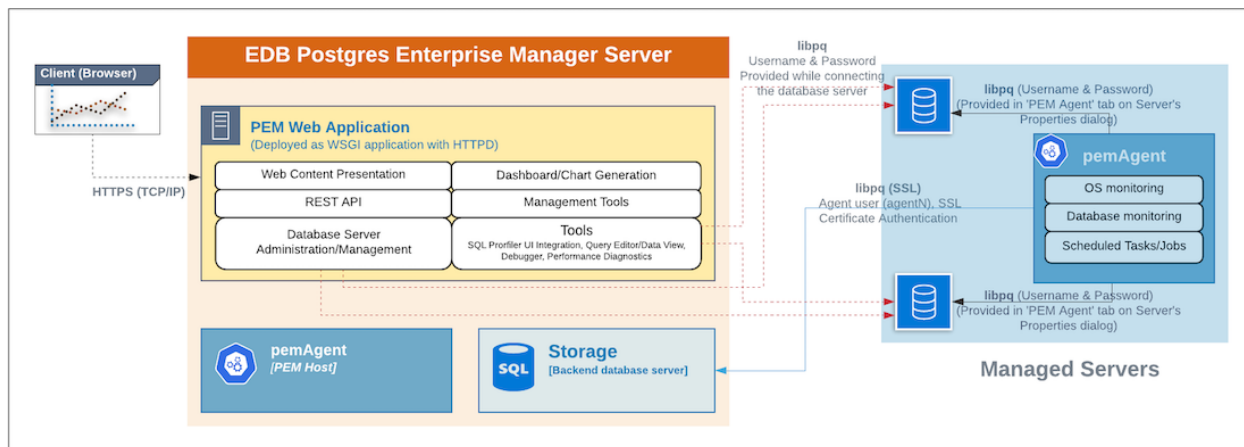


Fig. 3: PEM Agent

PEM Agent

The PEM Agent is responsible for the collection of monitoring data from the machine and operating system, as well as 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 - installed on the same system, or remotely - installed on other systems. It is also responsible for executing other tasks that may be scheduled by the user (for example, server shutdowns, SQL Profiler traces, user-defined jobs).

A PEM Agent is installed by default on the PEM Server along with the installation of the PEM Server. It is generally referred to as a PEM Agent on the PEM Host. Separately, the PEM Agent can also be installed on the other servers hosting the Postgres instances to be monitored 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.

Please note that 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 may optionally use SSL. The user 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 will 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

A list of PEM probes can be found [here](#).

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, processing by other PEM Agents can be enabled by adjusting the SNMP/SMTP and Nagios parameters of the PEM Agents.

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

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 the dashboards, the Postgres Log Analysis Expert, and Capacity Manager.

1.4 SQL Profiler plugin

You are not required to install the SQL Profiler plugin on every server, but you must install and configure the plugin on each server on which you wish to use the SQL Profiler. You may also want to install and configure SQL Profiler on un-monitored development servers. For ad-hoc use also, you may temporarily install the SQL Profiler plugin.

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](#).

SQL Profiler may be used on servers that are not managed through PEM, but 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 the *[SQL Profiler plugin](#)*.

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>

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 **Web Services** 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, it is a best practice to have PEM server and Apache web server on separate hosts.

2.1.2 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 Web Services and Database 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 non-production environments, it is a best practice to have PEM server and Apache web server on same hosts.

2.1.2.1 Using Failover Manager for high availability

Postgres Enterprise Manager (PEM) assists database administrators, system architects, and performance analysts in administering, monitoring, and tuning Postgres database servers.

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

This document provides detailed information to set up a high availability (HA) environment for PEM using Failover Manager.

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 needs to use the VIP address:

- The PEM Agent binding of the monitored database servers
- Accessing the PEM Web Client
- Accessing the Webserver services

2.1.2.1.1 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 the links above or refer to the instructions given on the

[EDB repos website](#) . You need to replace the USERNAME:PASSWORD with your username and password in the instructions to access the EDB repositories.

Ensure that the database server is configured to use `scram-sha-256` authentication method as the PEM Server configuration script does not work with `trust` authentication.

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

2. Configure the PEM Server on the primary server with initial configuration of type 1 (i.e. Web Services and Database):

edb_notranlate_0 For more detail on configuration types see,
[Configuring the PEM server to use Kerberos authentication](#).

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
```

2.1.2.1.2 Set up the primary node for streaming replication

1. Create the replication role using the following command.

```
$ /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 detailed information on configuring parameters for streaming replication refer to the [PostgreSQL documentation](#).

Note: The configuration parameters may be different for different versions of database server. You can email support at techsupport@enterprisedb.com for any help in 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 as below:

```
# 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
```

2.1.2.1.3 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 `.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
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 the `standby.signal` file on the standby nodes. The `postgresql.auto.conf` file content is as below:

```

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 krb_srvname=postgres target_session_attrs=any'

```

5. Edit the following parameter in `postgresql.conf` file on each of the standby nodes:

```
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, and set the following 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 \
              /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 ensures that webserver is configured on the standby and is disabled by default. It is enabled automatically at switchover by EFM.

Note: You need to keep the certificates in sync on master and standbys manually whenever the certificates are updated.

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

```
$ /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 & Stop Httpd & PEM Agent services if they are running on all replica nodes.

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

Note: At this point you should have a PEM Primary Server and two standbys that are ready to take over from the primary whenever needed.

2.1.2.1.4 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
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
```

(continues on next page)

(continued from previous page)

```

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
_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), and 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 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
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
```

(continues on next page)

(continued from previous page)

```

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		0/F7A3808	
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 get promoted as the primary node. PEM Agents automatically connect to the new primary node. You can replace the failed primary node with a new standby using this procedure.

2.1.2.1.5 Current limitations

The current limitations include:

- Web console sessions for the users are lost during the switch over.
- Per user settings done from the Preferences dialog are lost as they're stored in local configuration files on the file system.
- Background processes, started via Backup , Restore , and Maintenance dialogs, and their logs are not shared between the systems and are lost during switch over.

2.1.2.2 Connection pooling using pgBouncer

This document provides detailed information about using pgBouncer as a connection pooler for limiting the number of connections from the PEM Agent towards 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, please see the [Accessing the web interface](#).

This document uses the term **Postgres** to mean either the PostgreSQL or the Advanced Server database.

2.1.2.2.1 PEM server and agent connection management mechanism

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

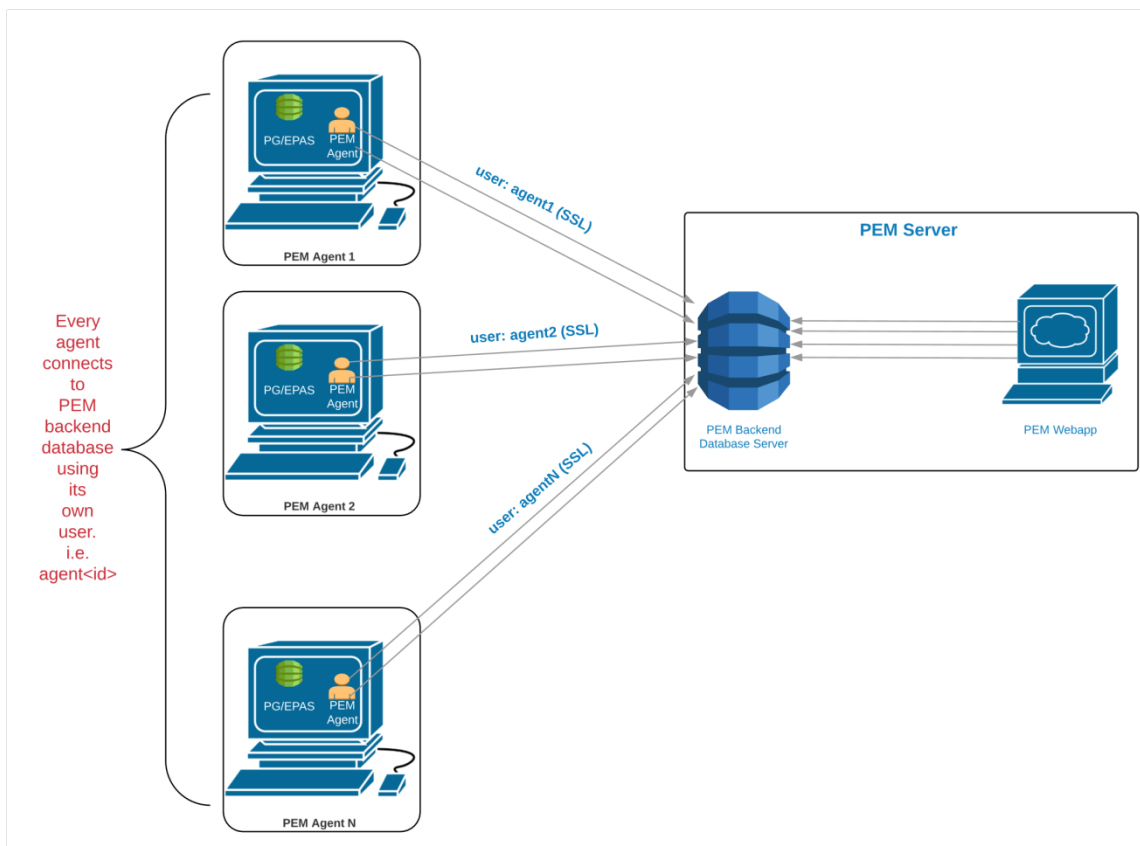


Fig. 1: Connecting to the PEM database without pgBouncer

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 the PEM database server.

- It uses an individual user identifier when connecting to the PEM database server.

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

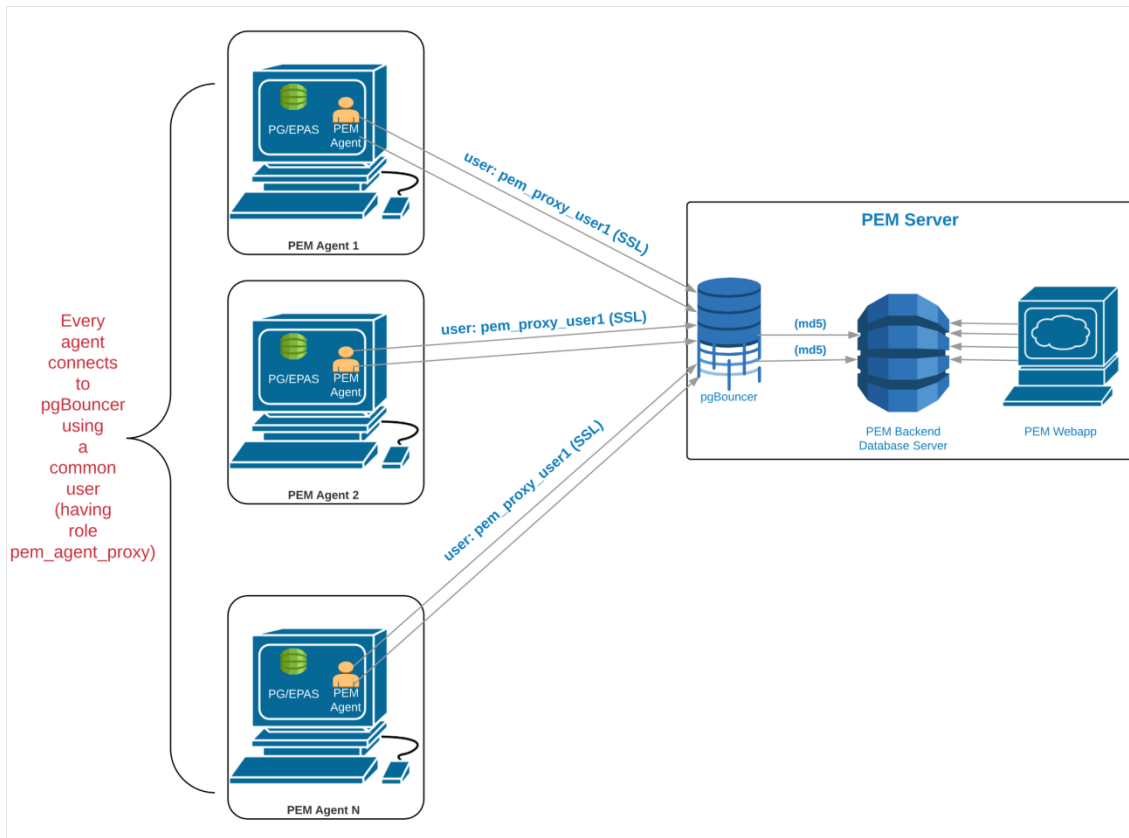


Fig. 2: Connecting to pgBouncer.

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

2.1.2.2.2 Preparing the PEM database server

You must configure the PEM database server to work with PgBouncer; the following example demonstrates the steps required to configure the PEM database server.

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

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

2. Create a user named `pem_admin1` (a non-super user) with `pem_admin` and `pem_agent_pool` role membership on the PEM database server. For example:

```
pem=# CREATE USER pem_admin1 PASSWORD ANY_PASSWORD LOGIN
CREATE ROLE;
CREATE ROLE
```

(continues on next page)

(continued from previous page)

```
pem=# GRANT pem_admin, pem_agent_pool TO pem_admin1;
GRANT ROLE
```

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

```
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. For example:

```
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. For example:

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

The function will create a user with the same name with a random password, and grant pem_agent and pem_agent_pool roles to the user. This 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; this will allow pgBouncer user to connect to the pem database using the md5 authentication method. For example:

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

local pem pgbouncer,pem_admin1 md5
```

2.1.2.2.3 Configuring PgBouncer

You must configure PgBouncer to work with the PEM database server. In our example, we will run PgBouncer as the enterprisedb system user. The following steps outline the process of configuring pgBouncer (version >= 1.9).

1. Open a terminal window and navigate into 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. For example:

```
$ 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 as follows:

```
[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
;; 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. Use the following command to create and update the `/etc/edb/pgbouncer1.9/userlist.txt` authentication file for PgBouncer.

```
pem=# COPY (
SELECT pgbouncer::TEXT, pgbouncer_password
```

(continues on next page)

(continued from previous page)

```

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 super user cannot invoke the PEM authentication query function `pem.get_proxy_auth(text)`. If the `pem_admin` user is a super user, you must add the password to the authentication file (*enterprisedb in the above 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`. For example:

```

$ 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. For example:

```

$ 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

```

2.1.2.2.4 Configuring the PEM agent

You can use an RPM package to install a PEM Agent; for detailed installation information, see the Installing agent on Linux .

Please note that PEM Agent which is responsible for sending SNMP notifications should not be configured with pg-Bouncer. For example, if the default PEM Agent installed along with PEM Server is used for SNMP notifications, then it should not be configured with pgBouncer.

Configuring a new PEM agent (installed via RPM)

After using an RPM package to install the PEM agent, you will need to configure it to work it 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 above 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 `/root/.pem` directory.

For example:

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

The keys will be used by the PEM agent to connect to the PEM database server as `pem_agent_user1` . It will also create an agent configuration file named `/usr/edb/pem/agent/etc/agent.cfg`.

You will find a line mentioning the agent-user to be used 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  
short_wait=10  
alert_threads=0  
enable_smtp=false  
enable_snmp=false  
enable_webhook=false  
max_webhook_retries=3
```

(continues on next page)

(continued from previous page)

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

Add a line for `agent_user` to be used for the agent. For example:

```
agent_user=pem_agent_user1
```

Update the port to specify the pgBouncer port. For example:

```
pem_port=6432
```

Update the certificate and key path locations. For example:

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

Please note: as an alternative, you can run the agent self registration script, but that will create a new agent id. If you do run the agent self-registration script, you must replace the new agent id with 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.

2.1.2.3 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 2FA authentication

2.1.2.3.1 Configuring the PEM server to use Kerberos authentication

- The Kerberos Server works with hostnames and not with ip addresses . To use single sign-on in PEM Server using Kerberos Authentication , following machines must be configured with hostnames using the DNS (REALM).

```
- Kerberos Server - PEM Server (PEM Web Server and PEM Backend Database Server) - Client machine
```

- For example the REALM on Kerberos Server is edbpem.org then Kerberos Server hostname can be set as Krb5server.edbpem.org , the PEM Server hostname as pem.edbpem.org and the client's hostname as pg12.edbpem.org .
- It is a convention to use the DNS domain name as the name of the realm.

Follow the steps to configure the Kerberos Authentication for the PEM Server:

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

- Install the Kerberos on the machine that functions as the authentication server.
- Install the PEM Server on the separate machine, for more information see Installing the PEM Server .
- Install the PEM Backend Database (Postgres/EDB Postgres Advanced Server) on same or different machine as PEM Server. For more information see the Installation steps on [EDB Docs Website](#) .

2. Adding 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>"
```

Where, 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 for both will be different.
-

3. Extract Key Tables from Kerberos server

- a. 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>"
```

- b. Copy the Key Tables from 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
```

c. On the PEM Server, move the Key Tables to required location and change the ownership as following:

```
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 to the Key Table for the PEM Web Application.
- NAME_OF_PEM_DB_FILE is the name specified to 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 (PostgreSQL/EDB Postgres Advanced Server).

4. Configure the PEM backend database server

- Add the Key Table location in 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 to the Key Table for the PEM Backend
↪Database Server.
- `DATA_DIRECTORY_OF_POSTGRES` is the path of the data directory of the installed
↪postgres (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>
```

Where,

```
- `POSTGRES_SERVICE_NAME` is the service name of Postgres (PostgreSQL/EDB Postgres,
↳ Advanced Server) database. For eg: 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. This utility is typically used to obtain the ticket-granting ticket, using a password entered by the user to decrypt the credential from the key distribution center (KDC). The ticket-granting ticket is then stored in the user's credential cache.
- You can view the details of the ticket using `klist` utility.

Note: The *Kerberos Client* must be installed on the PEM Server and the Client machine to use *kinit* and *klist* .

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

It will display 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 the `PEM_DB_HOST` in `config_setup.py` file and check the value of `PEM_AUTH_METHOD` is set to 'kerberos'. shell `$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/web/config_setup.py`
`PEM_DB_HOST=`<HOSTNAME_OF_PEM_SERVER>``
- Configure the `HOST` in `.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 will default to use the same authentication. To override the default, add the parameter `ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS` and set to `True` in `config_local.py` file.
- All the authenticated user principal will be appended with realm (USERNAME@REALM) and passed as database user name by default. To override the default, add the parameter `PEM_USER_KRB_INCLUDE_REALM` and set to `False` in `config_local.py` file.

- Restart the Apache server

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

- Edit the entries at the top in `pg_hba.conf` to use 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>
```

Where,

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

Note: If you are 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

- Open the low level Firefox configuration page by loading the `about:config` page. In the Search text 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 is protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with a comma. In the Search text 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 is protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with a comma. - Click 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: If you run into 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
```

The resolution is to add the additional encryption types to the keytab using `ktutil` or recreating the Postgres keytab with all crypto systems from AD.

2.1.2.3.2 Configuring the PEM server to use Windows Active Directory Domain Services for Kerberos authentication (SSPI)

- The Windows Active Directory Domain Services works with hostnames and not with ip addresses . To use single sign-on in PEM Server using Active Directory Domain Services , the following machines must be configured with hostnames using the DNS.

- Windows Server (Domain Controller) - PEM Server (PEM Web Server and PEM Backend Database Server) - Client machine

- For example the REALM on Windows Active Directory is `edbpem.internal` then Windows Server hostname can be set as `Krb5server.edbpem.internal` , the PEM Server hostname as `pem.edbpem.internal` and the client's hostname as `pg12.edbpem.internal` .

Follow the steps to configure the Kerberos Authentication for the PEM Server:

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

- Install the 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 it should be up and running.
- Install the PEM Server on the separate Linux machine, for more information see [Installing the PEM Server](#) .
- Install the PEM Backend Database (Postgres/EDB Postgres Advanced Server) on the same or different Linux machine as the PEM Server. For more information, see the Installation steps on [EDB Docs Website](#) .

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

- Create user in Active directory of Windows Server to map with the HTTP Service Principal for the PEM Web Application.
- a. Open Active Directory Users and Computers -> <DOMAIN_NAME> -> Users -> Right Click -> New -> User
- b. Enter the user details
- c. Give the password and make sure to deselect User must change password at next logon , and select User cannot change password and Password never expires .
- d. Review the user details
- e. On the PEMServer Web Properties , add the Users as member of the Domain Users group:

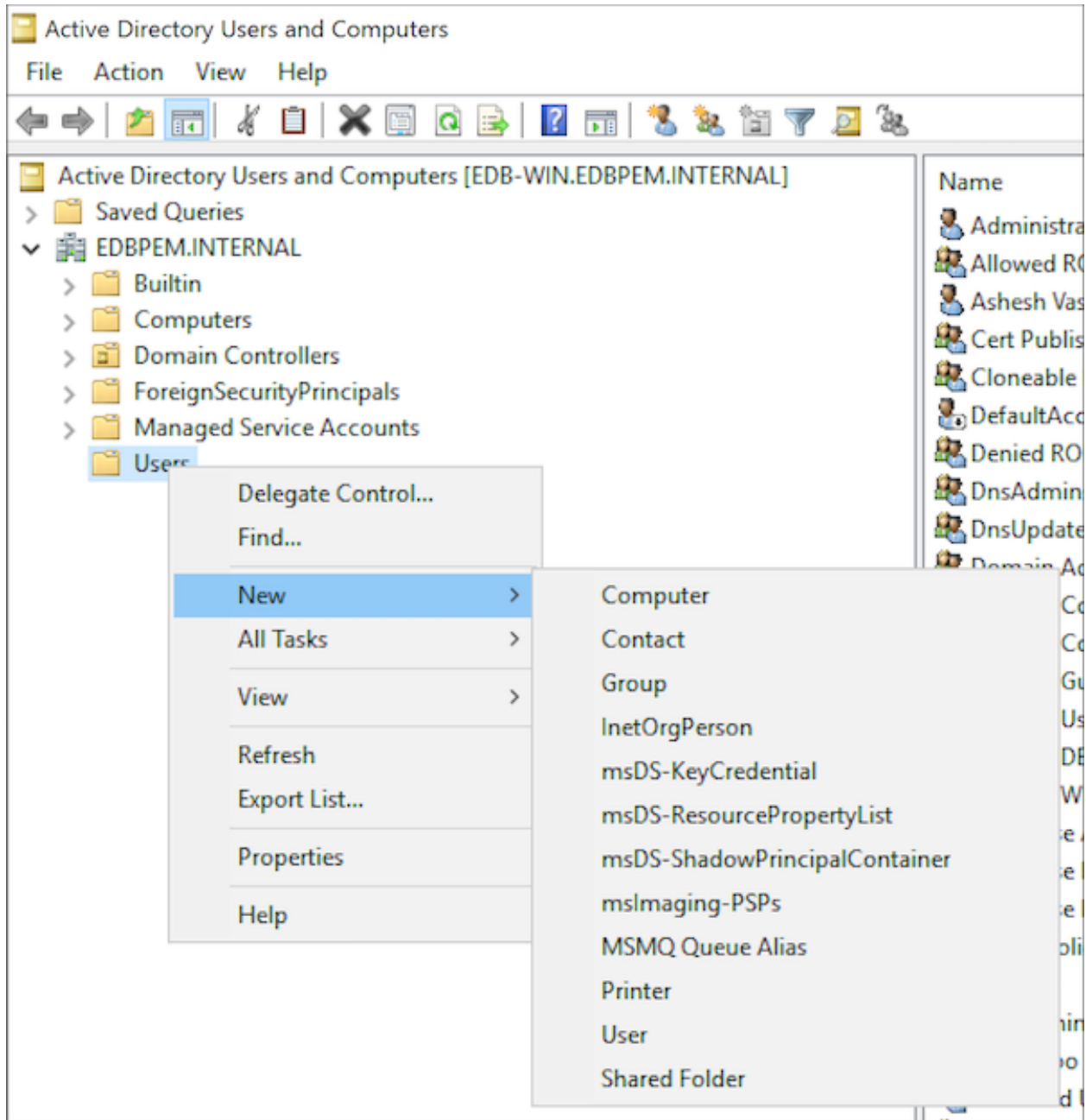
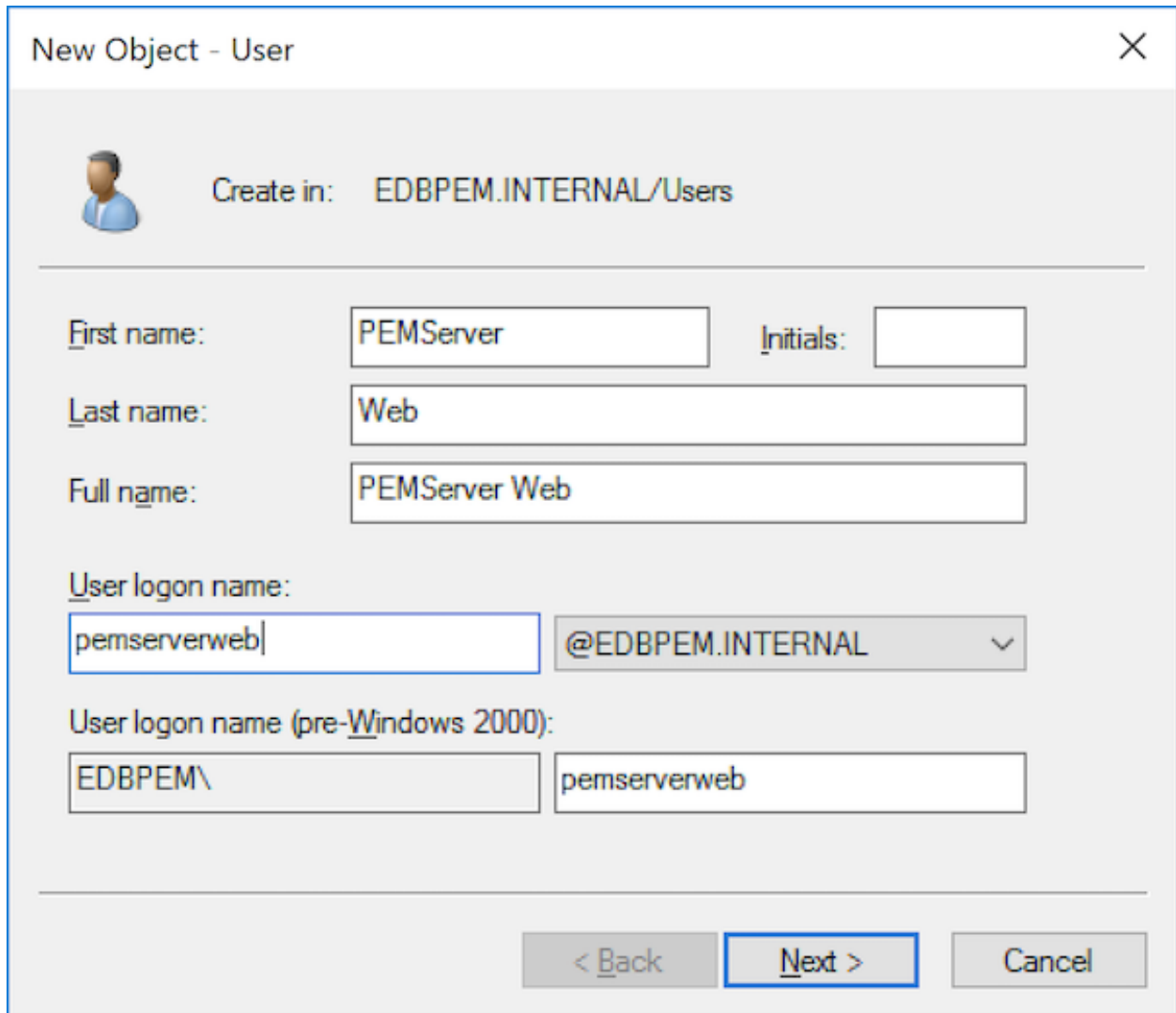


Fig. 3: Active Directory Users and Computers



New Object - User

Create in: EDBPEM.INTERNAL/Users

First name: PEMServer Initials:

Last name: Web

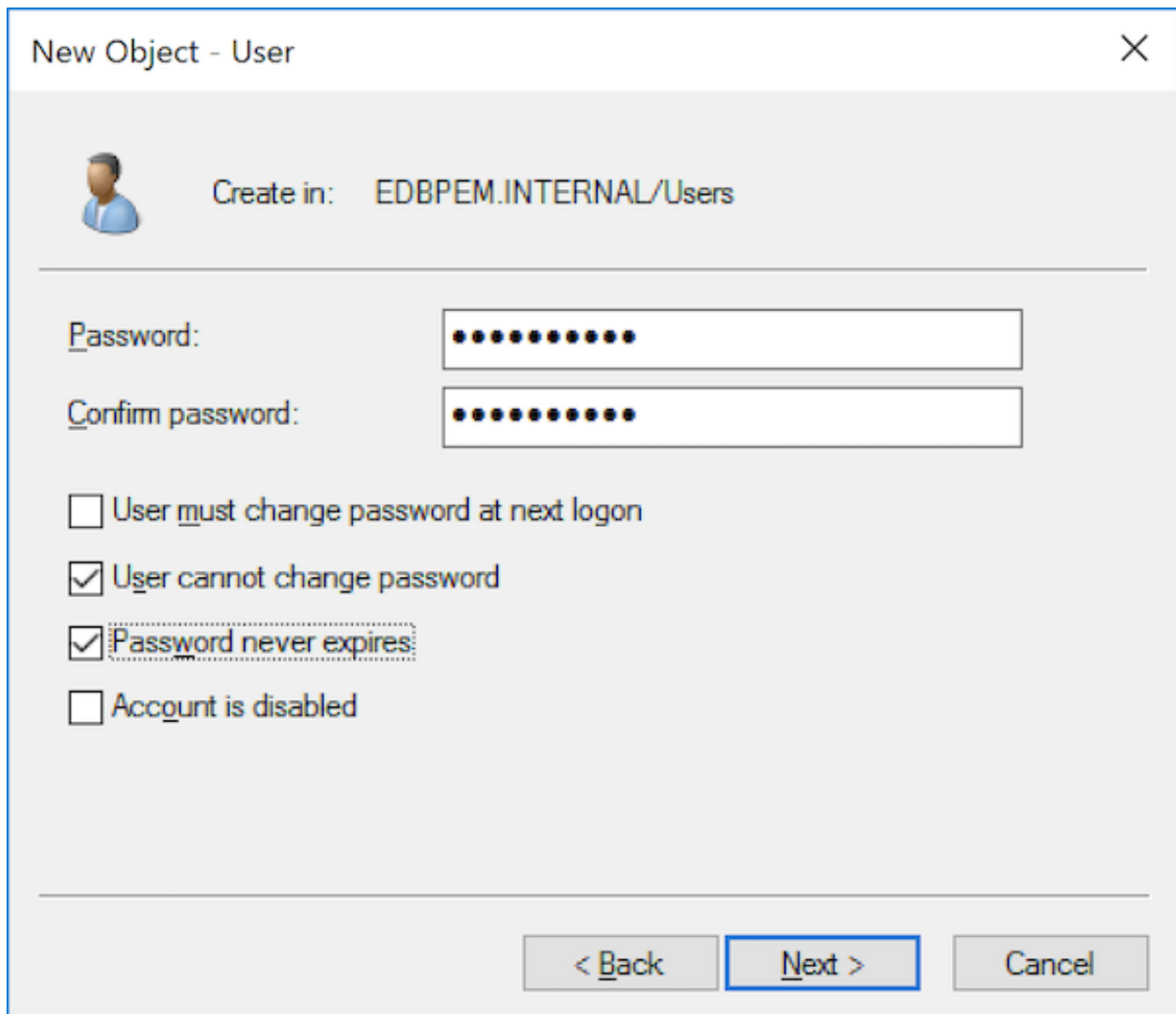
Full name: PEMServer Web

User logon name: pemserverweb @EDBPEM.INTERNAL

User logon name (pre-Windows 2000): EDBPEM\ pemserverweb

< Back Next > Cancel

Fig. 4: Active Directory Users and Computers Create User



The image shows a 'New Object - User' dialog box. At the top, there is a user icon and the text 'Create in: EDBPEM.INTERNAL/Users'. Below this, there are two password input fields: 'Password:' and 'Confirm password:', both containing masked characters. Under the password fields are four checkboxes: 'User must change password at next logon' (unchecked), 'User cannot change password' (checked), 'Password never expires' (checked), and 'Account is disabled' (unchecked). At the bottom, there are three buttons: '< Back', 'Next >' (highlighted with a blue border), and 'Cancel'.

New Object - User

Create in: EDBPEM.INTERNAL/Users

Password: [masked]

Confirm password: [masked]

☐ User must change password at next logon

☒ User cannot change password

☒ Password never expires

☐ Account is disabled

< Back Next > Cancel

Fig. 5: Active Directory Users and Computers set password

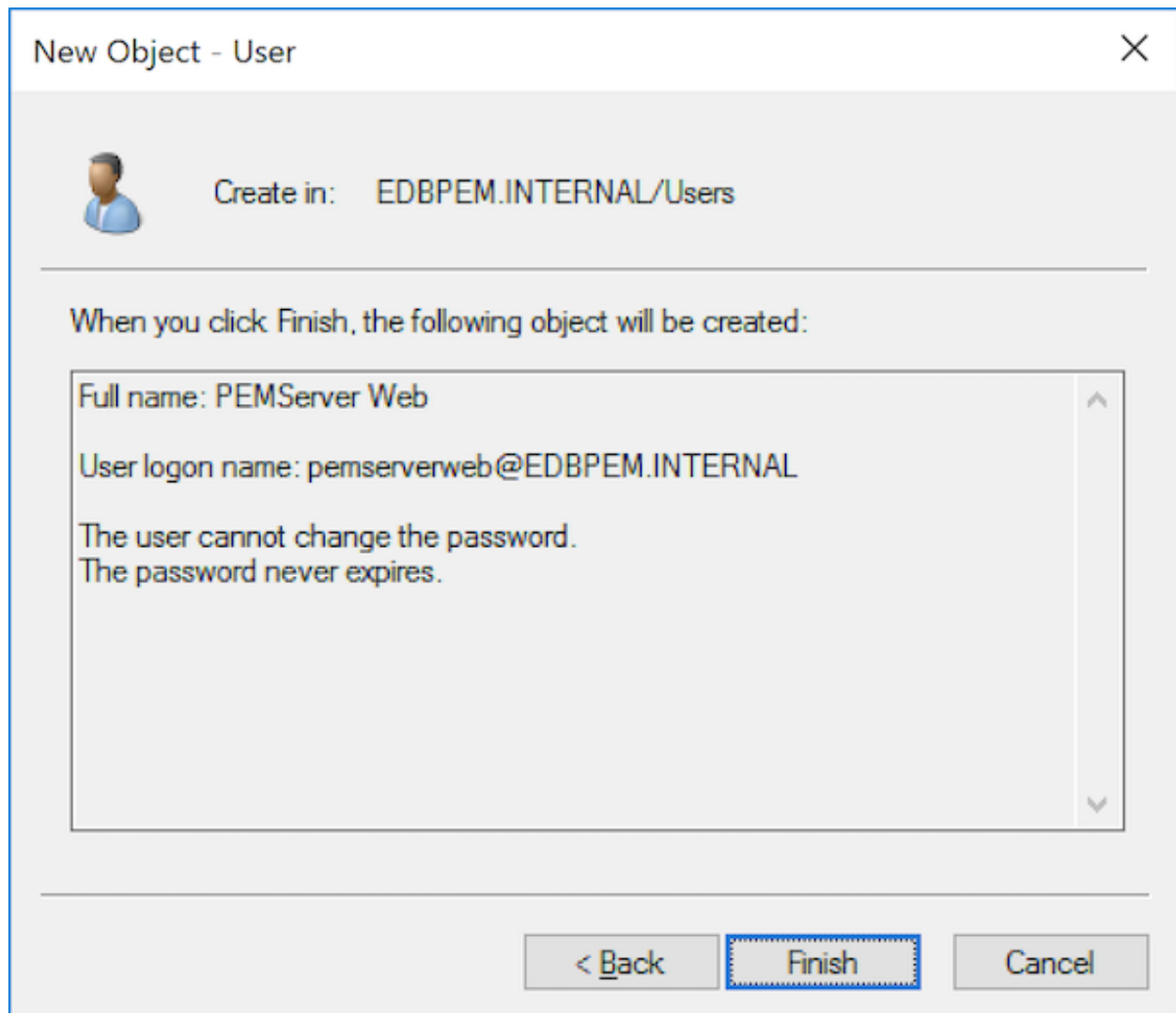


Fig. 6: Active Directory Users and Computers User Details

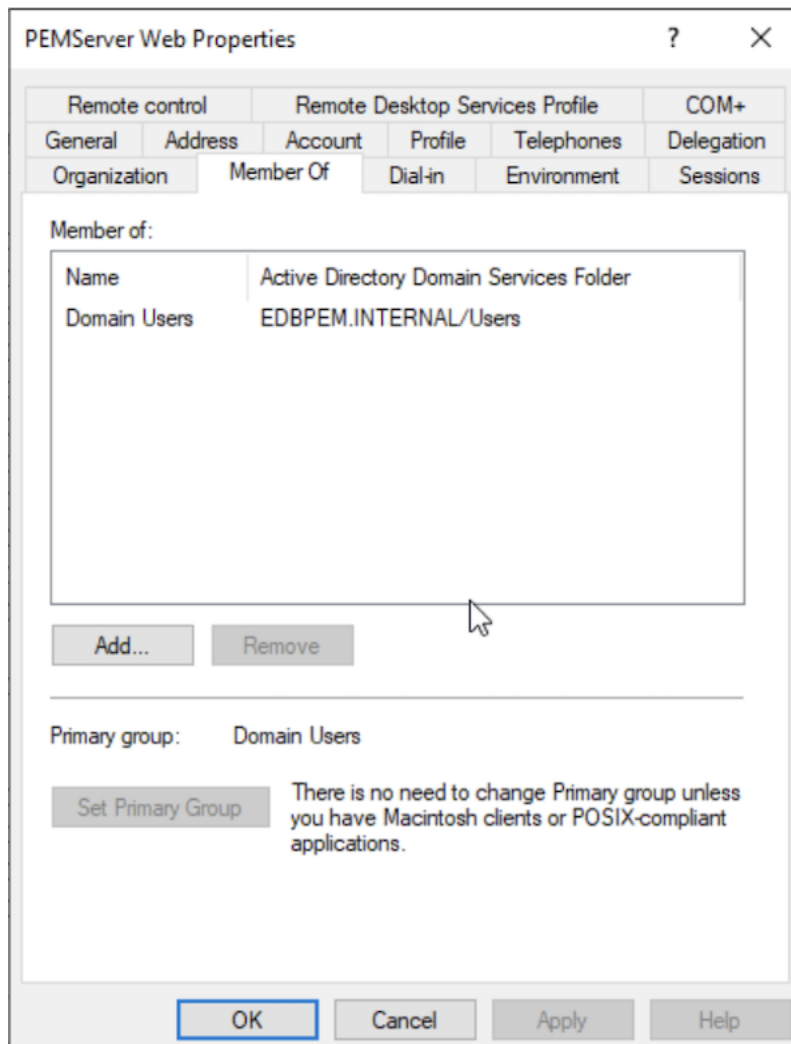


Fig. 7: PEM Server Web Properties

- Create user (for eg: `pemserverdb`) in Active directory of Windows Server to map with the Postgres Service Principal for the PEM Backend Database.

Repeat the steps from a to e as above.

3. Extract Key Tables from Active Directory

- Extract the Key Tables for the Service Principals and map with the respective domain users created in above step.
- a. Open Windows PowerShell as an Administrator user, create a Key Table for HTTP Service Principal mapping with user `pemserverweb` and a Key Table for Postgres Service Principal mapping with user `pemserverdb`.

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

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.edbpem.internal@EDBPEN.INTERNAL` is the hostname of the PEM Server (here `@EDBPEN.INTERNAL` means `@REALM`)
- `pemserverweb` is the user for the PEM Web Application
- `pemserverdb` is the user for the PEM Backend Database Server
- `EDBPEN.INTERNAL` is the domain of the target

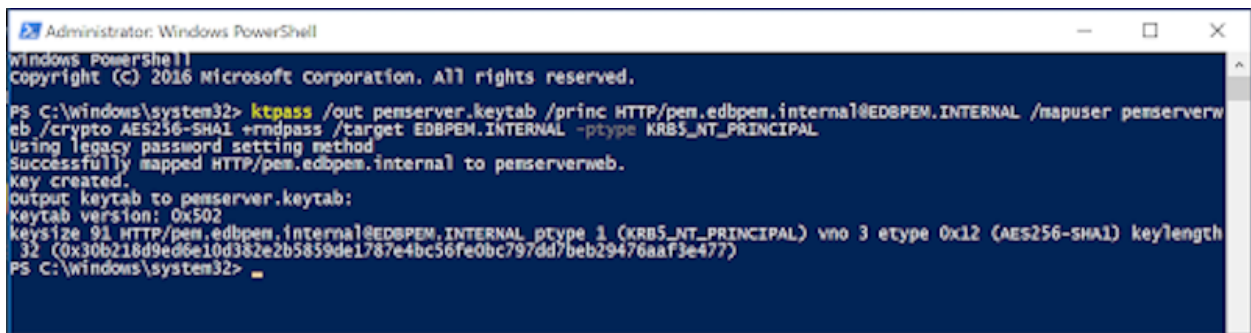


Fig. 8: Windows PowerShell ktpass

Note:

- The command line argument `+rndpass` will reset the password for the Domain User `pemserverweb` to a random password. - The `/target` option is optional.

- b. Add the Kerberos Support for the User accounts on the Accounts tab

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. 9: PEM Server Web Properties

Note: On the Account tab the *User logon name* shows *HTTP/pem.edbpem.internal@EDBPEM.INTERNAL* and *Delegation* tab is enabled for the *pemserverweb* user.

- c. Select **Trust this user for delegation to any service (Kerberos only)** option on the **Delegation** tab for the created users.
- d. Copy both the Key Tables to the PEM Server Host (or respectively to PEM Web Server and PEM Backend Database Server hosts if installed on different hosts.)
- e. On the PEM Server, move the Key Tables to the required location and change the ownership as following:

```
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 (PostgreSQL/EDB Postgres Advanced Server).

4. Configure the PEM backend database server

- Add the Key Table location in `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 (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

PEMServer Web Properties

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

Delegation is a security-sensitive operation, which allows services to act on behalf of another user.

☐ Do not trust this user for delegation
☒ Trust this user for delegation to any service (Kerberos only)
☐ Trust this user for delegation to specified services only

☒ Use Kerberos only
☐ Use any authentication protocol

Services to which this account can present delegated credentials:

Service Type	User or Computer	Port	Service Name

< >

☐ Expanded
 Add...
Remove

OK
Cancel
Apply
Help

Fig. 10: PEM Server Web Properties Delegation

```
systemctl restart <POSTGRES_SERVICE_NAME>
```

Where,

```
- `POSTGRES_SERVICE_NAME` is the service name of Postgres (PostgreSQL/EDB Postgres_
↳Advanced Server) database. For eg: 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. This utility is typically used to obtain the ticket-granting ticket, using a password entered by the user to decrypt the credential from the key distribution center (KDC). The ticket-granting ticket is then stored in the user's credential cache.
- You can view the details of the ticket using `klist` utility.

Note: The *Kerberos Client* must be installed on the PEM Server and the Client machine to use *kinit* and *klist*.

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

It will display 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=pem.edbpem.internal PEM_KRB_KTNAME=<PEM_INSTALLATION_DIRECTORY>/share/
↳pemserver.keytab <PEM_INSTALLATION_DIRECTORY>/bin/configure-pem-server.sh
```

- Configure the `PEM_DB_HOST` in `config_setup.py` file and check 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 the `HOST` in `.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 will default to use the same authentication. To override the default, add the parameter `ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS` and set to `True` in `config_local.py` file.
- All the authenticated user principal will be appended with realm (USERNAME@REALM) and passed as database user name by default. To override the default, add the parameter `PEM_USER_KRB_INCLUDE_REALM` and set to `False` in `config_local.py` file.

- Restart the Apache server

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

- Edit the entries at the top in `pg_hba.conf` to use 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>
```

Where,

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

Note: You cannot specify connection type as *hostgssenc* as windows does not 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 text 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 is protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with a comma. In the Search text 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 is protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with a comma. Click 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: If you run into 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
```

The resolution is to add the additional encryption types to the keytab using `ktutil` or recreating the Postgres keytab with all crypto systems from AD.

2.1.2.3.3 Configuring the PEM server to use two-factor authentication

- PEM supports two methods for 2FA:

- Email Authentication - Authenticator App (e.g. 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 this to true to enable the two-factor authentication. Default value is false.
<code>MFA_FORCE_REGISTRATION</code>	Set this to true to ask the users to register forcefully for the two-factor authentication methods at login. Default value is false.
<code>MFA_SUPPORTED_METHODS</code>	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.
<code>MFA_EMAIL_SUBJECT</code>	Set this to the subject of the email for email authentication. Default value is <code><APP_NAME> - Verification Code</code> .

Mail server configuration

To use the email authentication method, you need to configure Mail Server Settings.

PEM Server can send an email either using 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` , 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 refer to the [Flask-Mail documentation](#) .

Note: PEM SMTP alerts do not use this configuration.

2.1.2.4 Securing your deployment

This document provides information about security practices you should consider when configuring PEM. PEM functionality does not require you to enforce these practices; however, EDB recommends these practices to enhance the overall system's security.

PEM is dependent on third-party components from the vendor repository, including the Apache web server, OpenSSL, snmp++, libcurl, etc. To ensure these components are up to date, you should update your operating system and regularly apply security updates to avoid any security vulnerability. Without the most recent security patches, your system is potentially 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:

- Reduced exposure to cyberattacks
- Avoiding lost productivity
- Data protection from malware (like ransomware)
- Avoid worm infections that use security loopholes to spread over the network

2.1.2.4.1 Apache HTTPD security configurations

On a Windows system, Apache HTTPD is named PEM HTTPD, and the Apache httpd configuration file (`pme.conf`) is located in the `<Apache_Installation_Path>/conf/addons` directory. The ssl configuration file (`httpd-ssl-pem.conf`) is located in the `<Apache_Installation_Path>/conf/addons` directory.

On a Linux system, the apache httpd configuration file (`edb-pem.conf`) is located in the `<Apache_Installation_Path>/conf.d` directory and the SSL configuration file (`edb-ssl-pem.conf`) is located in the `<Apache_Installation_Path>/conf.d` directory.

Disable SSLv2 and SSLv3

You should disable SSL versions SSLv2, SSLv3, TLS 1, and TLS 1.1 as these versions are the most vulnerable and are affected by cryptographic concerns.

To disable the versions, please add the following command to apache httpd configuration file:

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

You need to restart the Web Server to apply the changes to the configuration file.

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

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

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

Secure httpd with SSL certificates

EDB recommends having an additional layer of SSL security for the web application.

By default, during PEM installation, PEM will generate and use self-signed certificates for Apache/HTTPD server. If you want to use your own SSL certificate for PEM, you must update the Apache configuration file.

On a Linux system, you need to open the Apache httpd configuration file (`edb-ssl-pem.conf`) in a text editor as 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.

There are two SSL Directives within the PEM VirtualHost section which you need to update:

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

For example, update:

```
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 Postgres Enterprise Manager. Follow the link shown below for the steps:

<https://www.enterprisedb.com/postgres-tutorials/how-replacing-httpd-self-signed-ssl-certificates-trusted-ca-signed-certificates>

Disable web server information exposure

EDB recommends disabling all web server signatures as part of web server security. The web server will expose a software signature; to disable the signature, add the following parameters to the Apache httpd configuration file. By default, PEM disables exposure of the information by adding the below parameters to the Apache httpd configuration file:

```
ServerTokens Prod
```

```
ServerSignature Off
```

The `ServerTokens` directive controls the server response header field, which is returned to the client. We recommend hiding 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 display of such information, we recommend disabling this directive in the Apache httpd configuration file.

You need to restart the web server to apply any changes to the Apache httpd configuration file.

Disable directory listing

The directory listing can allow an attacker to view complete directory contents. By default, the web server enables this option, and an attacker can discover and view any file. This listing could 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, you should disable the directory listing by setting the `Options` directive in the Apache `httpd` configuration file. By default, PEM disables the directory listing by setting the option below in the web server configuration file:

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

You need to restart the web server to apply the changes made to the configuration file.

Restrict the access to a network or IP address

Apache provides access control based on the client hostname or IP address. To view the application by specific IP address or network, a user can modify the Apache configuration file as shown below to provide your network address within the `Allow` directive:

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

PEM uses the `ALLOWED_HOSTS` configuration parameter in the application configuration file to provide the allowed hosts' list of IP addresses. The application configuration `config_local.py` file is located in `<PEM_INSTALLATION_PATH>/web`.

By default, PEM allows all the hosts to connect with the application.

For example:

You can set the range of IP addresses in the configuration file as below:

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

You can set the IP addresses to allow a host on a subnet level in the configuration file as below:

```
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 as below:

```
ALLOWED_HOSTS = ['127.0.0.1', '192.168.0.1']
```

You need to restart the web server to apply the changes to the application configuration file.

Cross-site tracing

There are two HTTP methods to debug the web server connections - TRACE and TRACK. When an HTTP TRACE request is sent to a web server that supports it, that server will respond, echoing the data passed to it, including any HTTP headers. We recommend disabling these methods within 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 does add the following lines to the Apache httpd configuration file:

```
RewriteEngine on
```

```
RewriteCond %{REQUEST_METHOD} ^ (TRACE|TRACK|OPTIONS)
```

```
RewriteRule .* - [F]
```

Run web server from a non-privileged user account

Running the Apache web server as a root user creates a security issue. We always recommend running the web server as a unique non-privileged user. This helps to secure other services running in the event of any security breaches.

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

On Linux, the Apache web server starts as the root user, but the daemon processes which PEM application runs as `pem` user. On Windows, the `WSGIDaemonProcess` directive and its features are not available. PEM HTTPD is installed as a service during the installation, and the user needs to be a member of the `Administrators` group for the service installation to succeed.

By default the Apache services are registered to run as the system user (the `LocalSystem` account).

Customize security HTTP headers in PEM web server

PEM contains its own configuration file to fix the following security issues. We recommend overriding the configuration only of `config_local.py` and not of `config.py`. The `config_local.py` is not present on the systems in most of the cases; hence users need to create it to override the application-level configurations. Please note that during a PEM upgrade, `config_local.py` will not be overwritten, but changes in `config.py` and `config_distro.py` will be overridden. Users need to remove `config_local.py` after uninstalling the PEM.

By default, `config_local.py` is located 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 may be able to use this input to inject harmful payloads that manipulate server-side behavior. The web applications typically don't know what domain they are deployed on unless specified in a configuration file during setup. When they need to know the current domain, for example, they may 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 & SQL injections.

X-Frame-Options

X-Frame-Options can be used to indicate if a browser should be allowed to render a page in an <iframe> tag. It was designed specifically to protect against clickjacking. PEM has a host validation `X_FRAME_OPTIONS` option to prevent such kind of 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 and provides a broader range of protection than the X-Frame-Options header (which it replaces). It is designed in such a way that website authors can whitelist individual domains from which resources (like scripts, stylesheets, and fonts) can be loaded, and also domains that are permitted to embed a page.

PEM has a host validation `CONTENT_SECURITY_POLICY` option to prevent such kinds of 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 response header (often abbreviated as HSTS) allows a web site or web application to tell browsers that it should only be accessed using HTTPS instead of HTTP. This option allows you to prevent a man-in-the-middle attack. The default is:

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

Note: Adding this parameter may cause problems if config is changed. Hence it is recommended to add this only after PEM installation is complete and tested.

X-Content-Type-Options

The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the MIME types advertised in Content-Type headers should not be changed and followed. This is a way to opt out of MIME type sniffing, or, in other words, 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 script into web pages viewed by other users. The HTTP X-XSS-Protection response to the header is a feature of Internet Explorer, Chrome, and Safari that stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. Although these protections are largely unnecessary in modern browsers when sites implement a strong Content-Security-Policy that disables the use of inline JavaScript (`'unsafe-inline'`) 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, you need to add the following options to the Apache configuration file:

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

A web server restart is required for configuration file changes to be applied.

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`.

This change requires an Apache service restart in order to take effect.

For more detailed information on `config.py` file you can see
PEM Online Help .

Cookie security

Cookies are small packets of data that a server can send to your browser to store configuration data. The browser automatically sends them along with all requests to the same server, so it's important to know how to secure cookies. There are multiple configuration options provided by PEM to make cookies secure which you can refer to in `config.py` but the three that follow are most important.

1. **SESSION_COOKIE_SECURE** - Setting the secure flag prevents the cookie from ever being sent over an unencrypted connection. It basically tells the browser to never add the cookie to any request to the server that does not use an encrypted channel. The browser will only add cookies to connections such as HTTPS. The default is:

```
SESSION_COOKIE_SECURE = True
```

2. **SESSION_COOKIE_HTTPONLY** - By default the content of cookies can be read via JavaScript. The HTTPOnly flag prevents scripts from reading the cookie. As the name HTTPOnly implies, the browser will only use the cookie in HTTP(S) requests. This prevents hackers from using XSS vulnerabilities to learn the contents of the cookie. For example, for the `sessionId` cookie it is never necessary to read the cookie with a client-side script, so for `sessionId` cookies, you can always set the HTTPOnly flag. The default is:

```
SESSION_COOKIE_HTTPONLY = True
```

3. **ENHANCED_COOKIE_PROTECTION** - When you set this option to `True` then a token will be generated according to the IP address and user agent. In all subsequent requests, the token will be recalculated and checked against the one computed for the first request. If the session cookie is stolen and the attacker tries to use it from another location, the generated token will be different, and in that case, the extension will clear the session and block the request. The default is:

```
ENHANCED_COOKIE_PROTECTION = True
```

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

This change requires an Apache service restart in order to take effect.

For more detailed information on `config.py` file you can see
PEM Online Help .

2.1.2.4.2 PEM application security configurations

Session timeout

Insufficient session expiration by the web application increases the exposure of other session-based attacks, as it allows time for the attacker to be able to reuse a valid session ID and hijack the associated session. The shorter the session interval is, the lesser the time an attacker has to use the valid session ID. We recommend setting the inactivity timeout for the web application to a low value to avoid this security issue.

Postgres Enterprise Manager provides a way to set the timeout value for a user session. When there is no user activity for a specified duration on the web console, PEM will log out the user from the web console. A PEM Administrator can set the length of time for inactivity. This value is application-wise, rather than for an individual user. To configure the timeout duration, modify the `USER_INACTIVITY_TIMEOUT` parameter in `config_local.py` file, located in the `<PEM_INSTALLATION_PATH>/web` directory. By default this functionality is disabled.

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

```
USER_INACTIVITY_TIMEOUT = 900
```

Note: The timeout value is specified in seconds.

This changes requires an Apache service restart in order to take effect.

For more detailed information on `config.py` file you can see
 PEM Online Help .

RestAPI header customization

You can customize the RestAPI token headers as per your requirements. The default values are not exposed by the `config.py` file; customize the following headers in the `config_local.py` file:

PEM_HEADER_SUBJECT_TOKEN_KEY

This configuration option will allow you to change the HTTP header name to get the generated token. By default, when the user sends a request to create a token, the server response will have an 'X-Subject-Token' header, which will contain the value of a newly generated token. If you want to customize the header name, then you can update the `config_local.py` file as shown:

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

Which will give you 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 will allow you to change the HTTP request header name through which you will send the token to the PEM server. By default, when the user sends a request to generate a token, the token header will be `X-Auth-Token`. If you want to customize the RestAPI request header name then you can update the `config_local.py` file as follows:

```
PEM_HEADER_TOKEN_KEY = 'Pem-Token'
```

Which will allow 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 will allow you to change the PEM RestAPI token expiry time after it is 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 as shown:

```
PEM_TOKEN_EXPIRY = 600
```

This changes requires an Apache service restart in order to take effect.

Role-based access control in PEM

Role-based access control (RBAC) restricts application access based on a user's role within 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 only allowed to access the information necessary to effectively perform their job duties. Roles in PEM are inheritable and additive rather than subtractive. In simple terms, as a PEM admin you need to grant the lowest level role to the user and then grant the roles which are required for the user to perform their respective tasks. 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 more detailed information on roles, you can see PEM Roles .

SQL/Protect plugin

Preventing a SQL injection attack is usually the responsibility of the application developer. The database administrator typically 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 provides a layer of security in addition to the standard database security policies by examining incoming queries for typical SQL injection profiles.

There are several different techniques used to perpetrate SQL injection attacks. A specific signature characterizes each technique. SQL/Protect examines queries for Unauthorized Relations, Utility Commands, SQL Tautology, Unbounded DML Statements. SQL/Protect gives the control back to the database administrator by alerting the administrator to potentially dangerous queries and blocking them.

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

For more detailed information about the SQL Profiler plugin, see the
PEM Online Help - SQL Profiler .

Password management

One security tip for PEM administrative users is to change your PEM login passwords to something new regularly. Changing your password avoids a number of dangers including:

- 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 ser

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 that should be used to run the script. You can either specify a non-root user or root user identity. If you do not specify a user, or the specified user does not exist, then the script will not execute. Restart the agent after modifying the file. If a non-root user is running **pemagent** , then the value of **batch_script_user** will be ignored, and the same non-root user used for running the **pemagent** will execute 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 will generate and use 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, follow the steps mentioned in the **Managing Certificates** section of the **PEM Administrators's Guide** :

Note: PEM does not support placing the SSL CA certificates at a custom location; you should not change the location of *ca_certificate.crt* and *ca_key.key* .

2.1.2.5 Web server installation options

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

2.1.2.5.1 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 Web Services 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, it is a best practice to have PEM server and Apache web server on separates 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 Web Services and Database 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 non-production environments, it is a best practice to have PEM server and Apache web server on same hosts.
