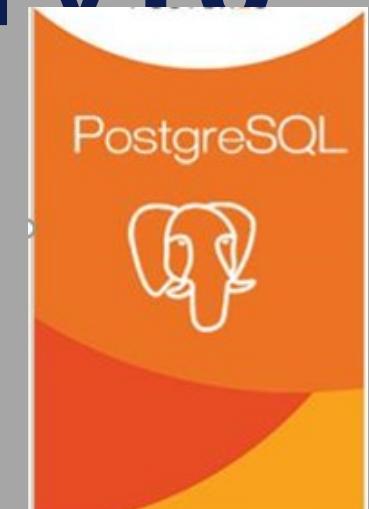




CURSO EDB Postgres Advanced Server DBA Advanced v10

OCTUBRE 2022



INSTRUCTOR

 <p>UNIVERSIDAD POLITÉCNICA DEL VALLE DE MÉXICO</p>	INGENIERO EN INFORMÁTICA. UNIVERSIDAD POLITÉCNICA DEL VALLE DE MÉXICO. ESTADO DE MÉXICO.
 <p>HEXAGON</p>	HEXAGON, LUCIAD FUSION, LIGHTSPEED, RIA. HEXAGON. CIUDAD DE MÉXICO.
 <p>Jenkins</p>	MASTER DEVOP'S, BACKEND. CIUDAD DE MÉXICO.
 <p>VM Edu Authorized Training Partner</p>	SCRUM FUNDAMENTALS. CIUDAD DE MÉXICO.

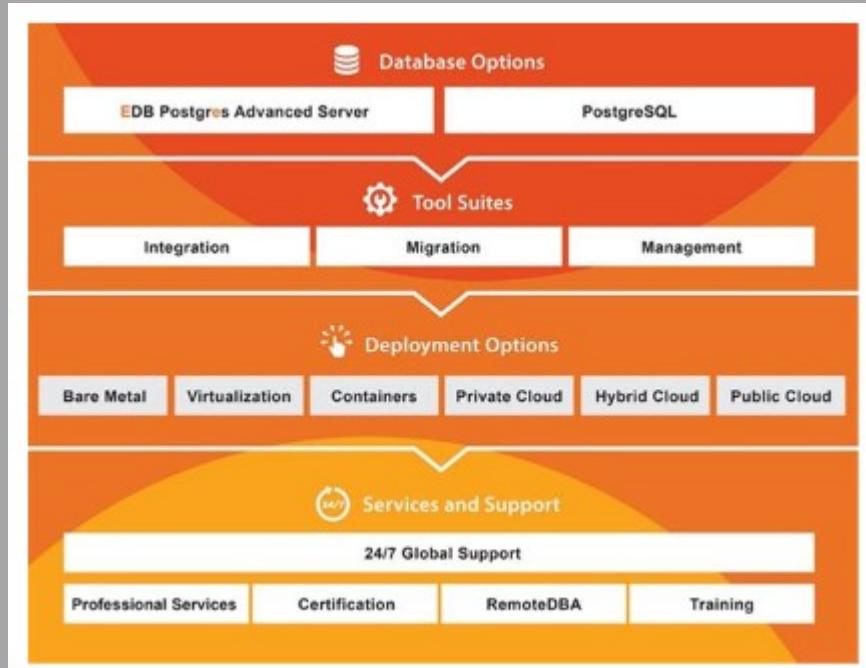
AGENDA.

- Introducción.
- Monitoreo de Base de Datos.
- SQL Tuning.
- Performance Tuning.
- Benchmarking.
- Extensiones.
- Tabla de Particiones.

AGENDA.

- Connection Pooling.
- Clonación.
- High Availability y Replicación.
- Seguridad a nivel Base de Datos.
- Mejores Practicas.

INTRODUCCIÓN.



EDB incluye todo lo necesario para que el cliente pueda integrar todos los servicios ofrecidos en la suscripción.

Desde servicio global 7x24 , entrenamiento Soporte online y certificaciones.

Opciones de deployment desde la nube hasta esquemas híbridos.

Herramientas de migración, administración que pueden ser descargadas desde el sitio web

Así como las opciones de ir por EDB o PostgreSQL según las necesidades empresariales.

INTRODUCCIÓN.

1. EDB Postgres Advanced Server.

- Basado en el estándar open-source PostgreSQL.
- Incluye mejoras en el performance, seguridad, y compatibilidad con Oracle.
- Incluye herramientas para desarrolladores y DBA (features).
- 24x7 Soporte Enterprise , integraciones y apoyo con herramientas de migración

2. PostgreSQL.

- Sin duda la base de datos open source más utilizada por al ámbito profesional.
- 24x7 EDB Soporte Estandar integraciones y apoyo con herramientas de migración.

EDB POSTGRES PLATFORM - TOOLS SUITES.

Integration

Intercambio de datos en tiempo real con diferentes herramientas de administración.

- EDB Postgres Data Adapters.
- EDB Postgres Replication Server.
- EDB Postgres XA Support”.

Migration

Posibilidad de migración de datos de otras marcas caras en el Mercado de una forma sencilla gracias a las heramientas:

- EDB Postgres Migration Assessment with services.
- Engagement.
- EDB Postgres Migration Took

Management

- Uso de herramientas de mission crítica para monitoreo, tuning, respaldos y alta disponibilidad.
- EDB Postgres Enterprise Manager.
- EDB Postgres Failover Manager.
- EDB Postgres Backup and Recovery.

EDB POSTGRES PLATFORM - DEPLOYMENT OPTIONS.

Bare Metal

- Mejor opción para maximizar la sencillez y control de una base de datos.
- Esta opción de deploy ya ha sido testeada y certificada en la mayoría de sistemas operativos.

Virtualización

- La opción más consolidada para ahorrar costos en inversión de hardware.
- Soportada por Vmware, KVM y otras plataformas de virtualización.

Contenedores

- Provee unidades pequeñas en forma granular para asegurar el óptimo uso de recursos.
- EDB es desarrollado en base a componentes de forma inteligente que hace que su agrupación en contenedores sea más sencilla.

EDB POSTGRES PLATFORM - DEPLOYMENT OPTIONS.

Private Cloud

- La opción on premise que ofrece una base de datos customizable, escalable y aprovisionamiento self service.
- OpenStack (IaaS).
- OpenShift(PaaS).
- Cloud Foundry (PaaS)

Public Cloud

- Flexibilidad y control sobre costos.
- Incluida en Amazon AWS, AZURE, Google .

Hibrid Cloud

- EDB Postgres Ark cloud DBaaS.
- Intercambio de datos y replicación.
- Compatible diferentes tipos de nube.

EDB POSTGRES PLATFORM - SERVICES AND SUPPORT.



- **24 x 7 Global Support.**
- **Servicios Profesionales**

Apoyo en inicio, instalación y entrenamiento.

- **Certificaciones.**

En base al entrenamiento en los cursos podrás acceder a realizar un examen de certificación con reconocimiento internacional como DBA de Postgres.

- **RemoteDBA.**

Apoyo de DBA online para implementaciones.

- **Training.**

- Cursos de talla mundial con contenido exclusivo.
- Curso de foundation y advanced para cubrir todos los temas de implementación hasta tuning, administración y migración.

EDB POSTGRES PLATFORM - FACTS ABOUT POSTGRESQL.

Innovación tecnológica en defensa y seguridad
DECSEF

- La base de datos más utilizada a nivel mundial de open source.
- Diseñada para extensibilidad y customización.
- Cumple la normatividad ANSI/ISO SQL Support.
- Desarrollada por más de 20 años.
- Universidad de Postgres (1986-1993).
- Postgres95 (1994-1995).
- PostgreSQL (1996- Actual).



EDB POSTGRES PLATFORM - FACTS ABOUT POSTGRESQL.

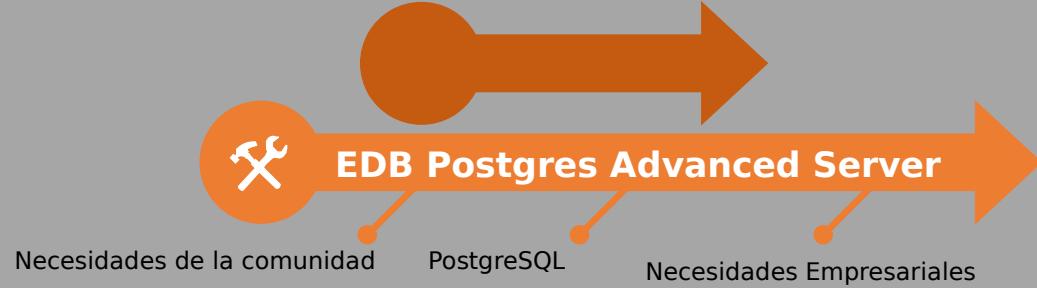
Innovación tecnológica en defensa y seguridad
DECSEF

PostgreSQL es una potente base de datos relacional Open Source.

- Hoy en día es una de las BBDD Open Source más empleadas del mundo.
- Posee una licencia *PostgreSQL License* a las licencias del MIT.
- Está diseñada de manera modular y extensible.
- Está escrita en C.
- Sigue el estándar ANSI-ISO:2008 por lo que es SQL estándar.
- Desarrollada desde hace más de 20 años.
- Décima generación de producto.
- Multiplataforma.



EDB POSTGRES PLATFORM - FACTS ABOUT POSTGRESQL.



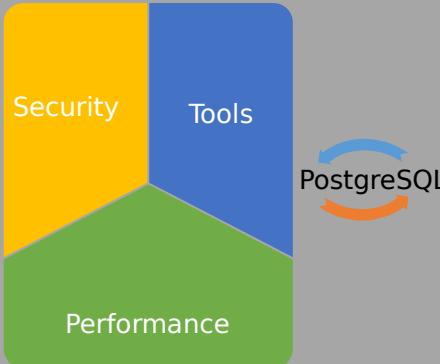
- EDB Postgres Advanced esta construido bajo los ladrillos de PostgreSQL Foundation.
- PostgreSQL ofrece mejoras en Performance, Compatibilidad y Seguridad.
- Bajo costo.



EDB POSTGRES PLATFORM - FACTS ABOUT POSTGRESQL.

- Security: Políticas de passwords, Auditoria, Row Level Security, Protección ante ataques SQL , server side protection.

EDB Postgres



- Tools: Herramientas de monitoreo y administración.
- HA failover Management.
- Multi-master replicación.
- Oracle, SQL Server, MySQL hacia Postgres.

- Performance: Ajuste de CPU, IO, cargas de trabajo.
- Particionamiento rápido.
- SQL Profiler.
- Indexación rápida.
- Dyna Tune Upgrades de memoria.
- Detección de cuellos de botella SQL.



EDB POSTGRES PLATFORM - FACTS ABOUT POSTGRESQL.



Compatibilidad con Oracle.

- Fácil migración de Oracle.
- Soporte a PL/SQL.
- Objetos definidos compatibles en Postgres.
- Paquetes de funciones.
- Herramientas Oracle EDB Loader, EDB Plus, EDB Wrap.



EDB POSTGRES PLATFORM- LÍMITES

LÍMITE	VALOR
Tamaño máximo de BBDD	Ilimitado
Tamaño máximo de tabla	32 TB
Tamaño máximo de fila	1,6 TB
Tamaño máximo de campo	1GB
Máximo número de filas por tabla	Ilimitado
Máximo número de columnas por tabla	250 – 1600 dependiendo del tipo de columna
Máximo número de índices por tabla	Ilimitado
Máximo número de columnas por índice	32



EDB POSTGRES PLATFORM - NOMENCLATURA.



TÉRMINO INDUSTRIAL	TÉRMINO POSTGRESQL
Instancia	Cluster
Tabla o índice	Relación
Línea o fila	Tupla
Columna	Atributo
Bloque de datos	Página (cuando el bloque está en disco)
Página	Buffer (cuando el bloque está en memoria)

- Corchetes ([abc]) indican partes opcionales.
- Conjuntamente llaves ({ abc }) y pipe () indican que debes elegir una alternativa.
- Puntos suspensivos (...) significa que el elemento puede repetirse.
- Comandos SQL van precedidos de prompt =>, y comandos de shell de prompt \$.



EDB POSTGRES PLATFORM - PREPARACIÓN DE SAMPLE

Prepare a Sample Database

- Dentro del material de studio proveído por EnterpriseDB tenemos un script file llamado “edbstore.sql” el cual puede ser ejecutado usando el comando:
edb-psql

Pasos:

- Bajamos el archive llamado edbstore.sql y lo guardamos en el home del Usuario enterpriseDB user.
- Login como enterpriseDB OS user.
- Ejecutamos el comando edb-psql con la opción -f sobre el archive edbstore.sql para instalar todo lo que vamos a requerir para el entrenamiento.
- edb-psql -p 5444 -f edbstore.aql -d edbstore -U edbuser.

*** Nota - password es edbuser**



EDB POSTGRES PLATFORM - PREPARACIÓN DE SAMPLE

Prepare a Sample Database

- Para comprobar que todo esta bien debemos de conectarnos a la base de datos *edbstore* usando el Usuario *edbuser*.
- Verificamos la existencia de los objetos.

```
edbstore=> \d
          List of relations
 Schema |      Name       |   Type   | Owner
-----+---------------+----------+-------
 edbuser | categories    | table    | edbuser
 edbuser | categories_category_seq | sequence | edbuser
 edbuser | cust_hist    | table    | edbuser
 edbuser | customers    | table    | edbuser
 edbuser | customers_customerid_seq | sequence | edbuser
 edbuser | dept         | table    | edbuser
 edbuser | emp          | table    | edbuser
 edbuser | inventory    | table    | edbuser
 edbuser | job_grd      | table    | edbuser
 edbuser | jobhist      | table    | edbuser
 edbuser | locations     | table    | edbuser
 edbuser | next_empno   | sequence | edbuser
 edbuser | orderlines   | table    | edbuser
 edbuser | orders        | table    | edbuser
 edbuser | orders_orderid_seq | sequence | edbuser
 edbuser | products      | table    | edbuser
 edbuser | products_prod_id_seq | sequence | edbuser
 edbuser | reorder       | table    | edbuser
 edbuser | salesemp      | view     | edbuser
 edbuser | vw_mat_cust   | materialized view | edbuser
 edbuser | vw_emp_details | view     | edbuser
(21 rows)
```



EDB POSTGRES PLATFORM - MONITOREO DE BASE DE DATOS



- El monitoreo de base de datos consiste en capturar y grabar eventos dentro de una base de datos.
- Esta información ayuda a los DBA's para detectar, identificar y reparar errores potenciales en la base de datos.
- Al realizar el monitoreo por medio de estadísticas hace más fácil su lectura e identificación de la salud del sistema.
- Existen muchas herramientas de monitoreo que analizan la salud y desempeño de una base de datos.



EDB POSTGRES PLATFORM – PREPARACIÓN DE SAMPLE



Estadísticas de base de datos

Las estadísticas de la base de datos se almacenan en catálogos internos y se almacena:

- Current running sessions.
- Running SQL.
- Locks.
- DML counts.
- Row Counts.
- Index usage

Stats Collector

Pg_catalog
Statistics Tables



EDB POSTGRES PLATFORM - STATISTIC

- Stat Collector es el proceso que colecta y reporta la actividad en la base de datos.
- Stats Collector agrega algo de overhead en la ejecución de una query.
- Stats Parametros:
 - track_counts - Control de tabla e indexaciones.
 - track_activities- Habilita el monitoreo del comando en ejecución.
- El stats collector usa archivos temporales en el subdirectorio pg_stat-tmp
- Estadísticas permanentes son almacenadas en el esquema pg_catalog dentro del subdirectorio global
- TIP: El parámetro stats_temp_directory puede ser igualado al mismo espacio de la memoria RAM destinada al sistema.



EDB POSTGRES PLATFORM - STATISTICS TABLES

- ***pg_catalog*** esquema contiene un set de tablas, vistas y funciones sobre las cuales guardar y reportear las estadísticas de la base.
- ***pg_class* y *pg_statst*** tablas de catalogos almacenan las estadísticas.
- ***pg_stat_database*** vista puede ser usada para ver métricas de información avanzada de la base y ver medidas.
- ***pg_stat_bgwriter*** muestra las estadísticas de background del writer así como información de los checkpoints.
- ***pg_stat_user_tables*** muestra información de las actividades de una tabla tales como inserts, updates, deletes, vacuum etc.
- ***pg_stat_user_indexes*** muestra información de las indexaciones usadas por las tablas.
- ***pg_stat_progress_vacumm*** muestra una columna por cada vacuum worker process que se este ejecutando.



EDB POSTGRES ENTERPRISE MANAGER-PEM



Una solución completa que combina la potencia de tres herramientas.

Consola WEB fácil uso

Fácil instalación

Funciona con PostgreSQL y con EDB Postgres Advanced Server



Monitor



Administración



Tuneo

POSTGRES ADVANCE SERVER.

Postgres Enterprise Manager (PEM).

- Está diseñado para administrar, monitorear y tunear EDB Postgres Advanced Server.
- Puede ser usado para Administrar multiples clusters.
- Soporta tanto a PostgreSQL como a EDB Postgres Advanced Server.
- PEM es una interfaz web similar a PGADMIN pero esta es nativa de EDB.

POSTGRES ADVANCE SERVER.

Postgres Enterprise Manager (Características).

- Administra multiples EDB Postgres Advanced Server y PostgreSQL instancias.
- Ofrece un Dashboard Global.
- Colecta y muestra información de todo el cluster.
- Optimizado para SQL , Capacidad, Auiditoría.
- Se compone de PEM Server y PEM Agent.

POSTGRES ADVANCE SERVER.

Postgres Enterprise Manager (Navegar)

- Puede ser abierto a través de un navegador.
- WEB-Based-Tool.
- Similar a PGADMIN pero esta es interna de EDB.

EDB POSTGRES ENTERPRISE MANAGER-PEM



Monitor

- Misión crítica y recolección de estadísticas.
- Alertas vía SMNP o SMTP.
- Creación de Dashboards.
- Monitoreo de la replicación.



Tuneo

- SQL Profiles.
- Sugerencias de indexación.
- Wizard de Tuning.

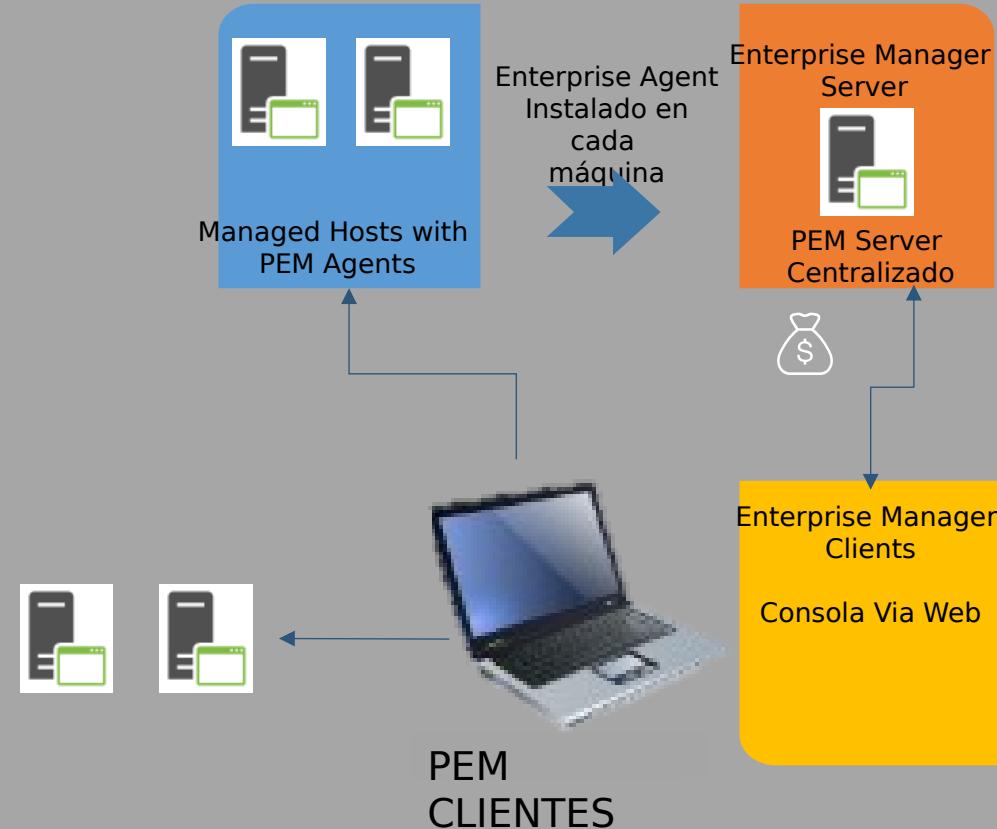


Administración

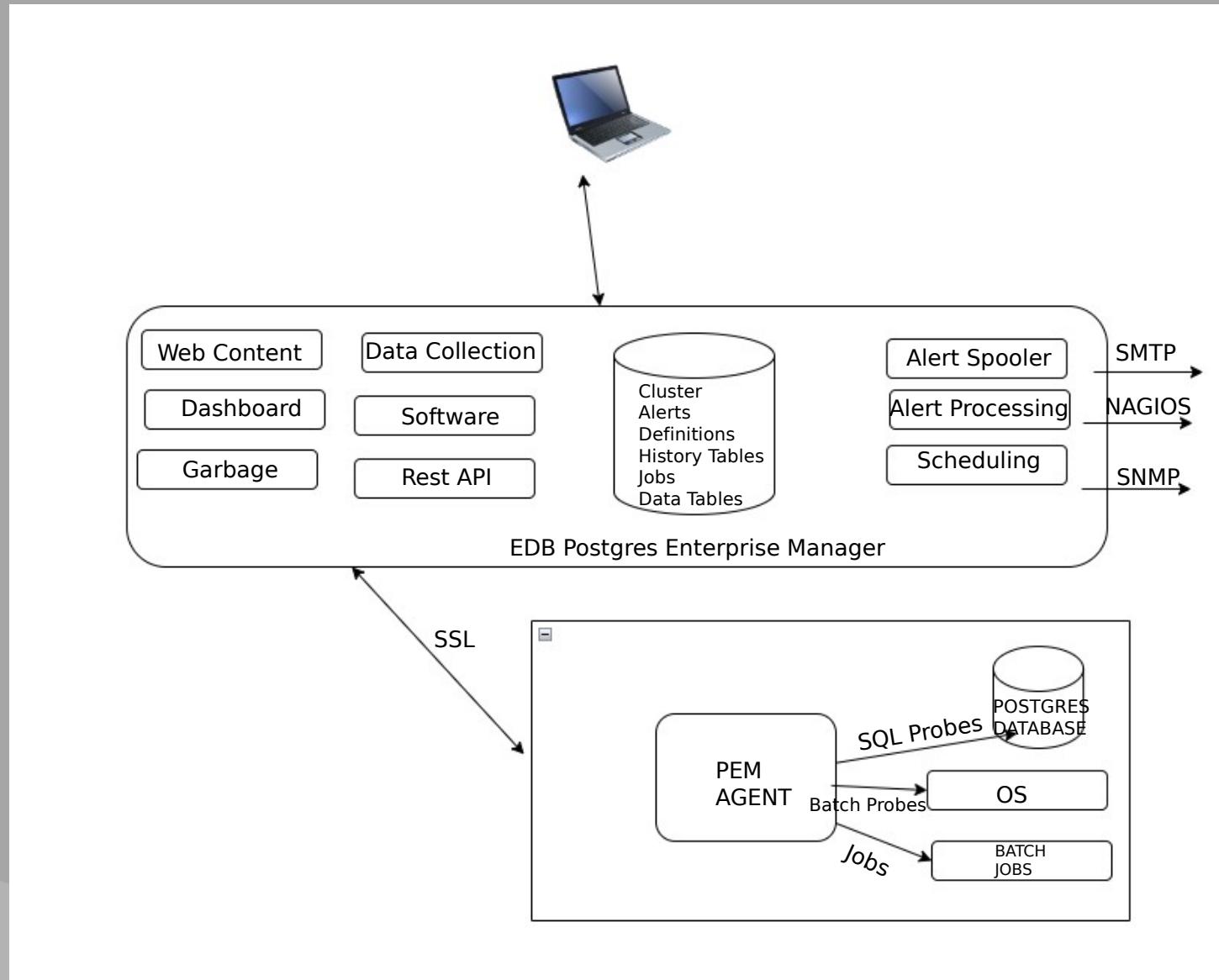
- CRUD operaciones en todos los objetos de la base de datos.
- Operaciones en “bulk” a través de múltiples servidores.
- Capacidad de Planeación y administración de recursos.
- GUI customizable , gráficos, tablas.

PEM ARCHITECTURE.

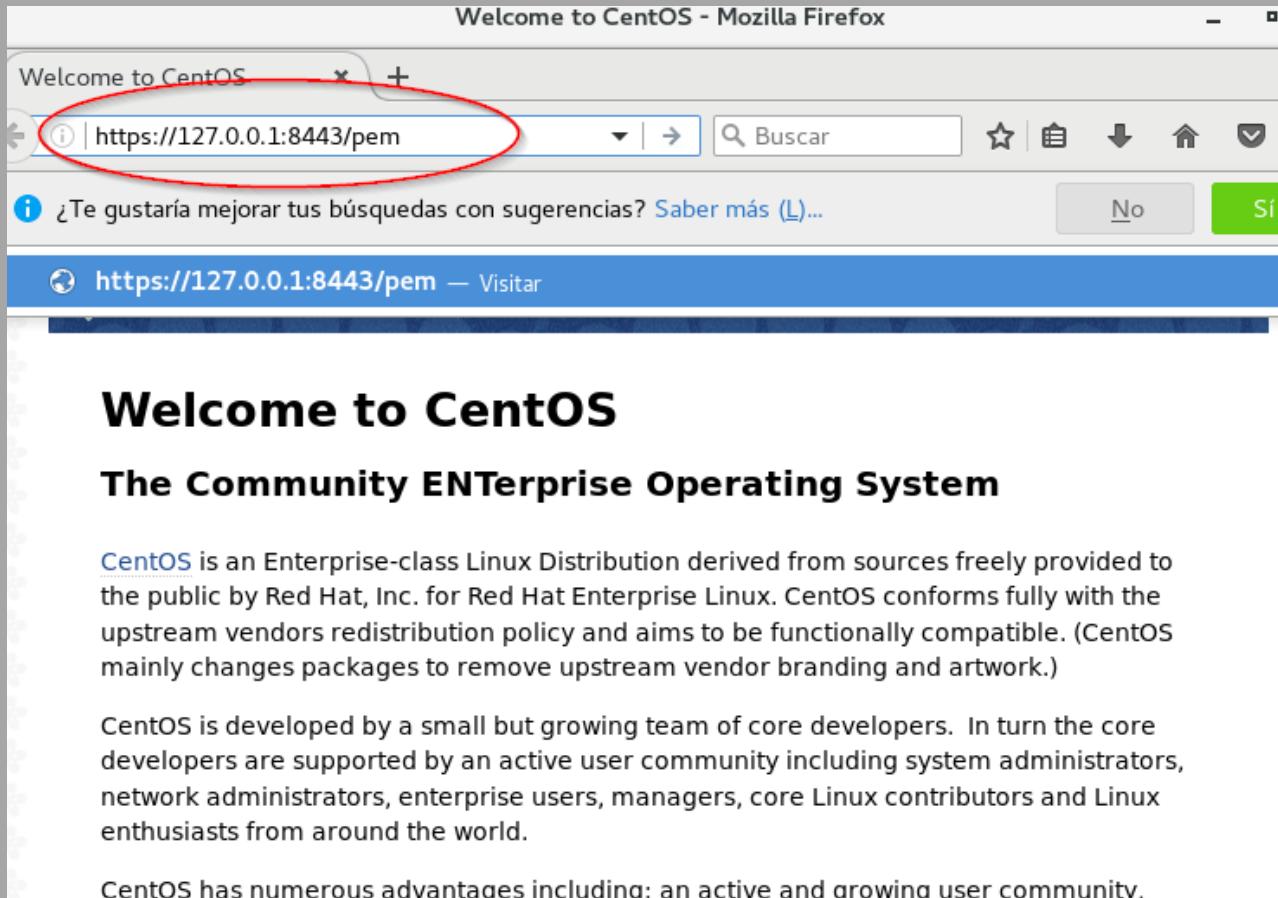
- Una arquitectura flexible.
- Fácil de implementar.
- Administra.
- Tuning.
- Dashboards.



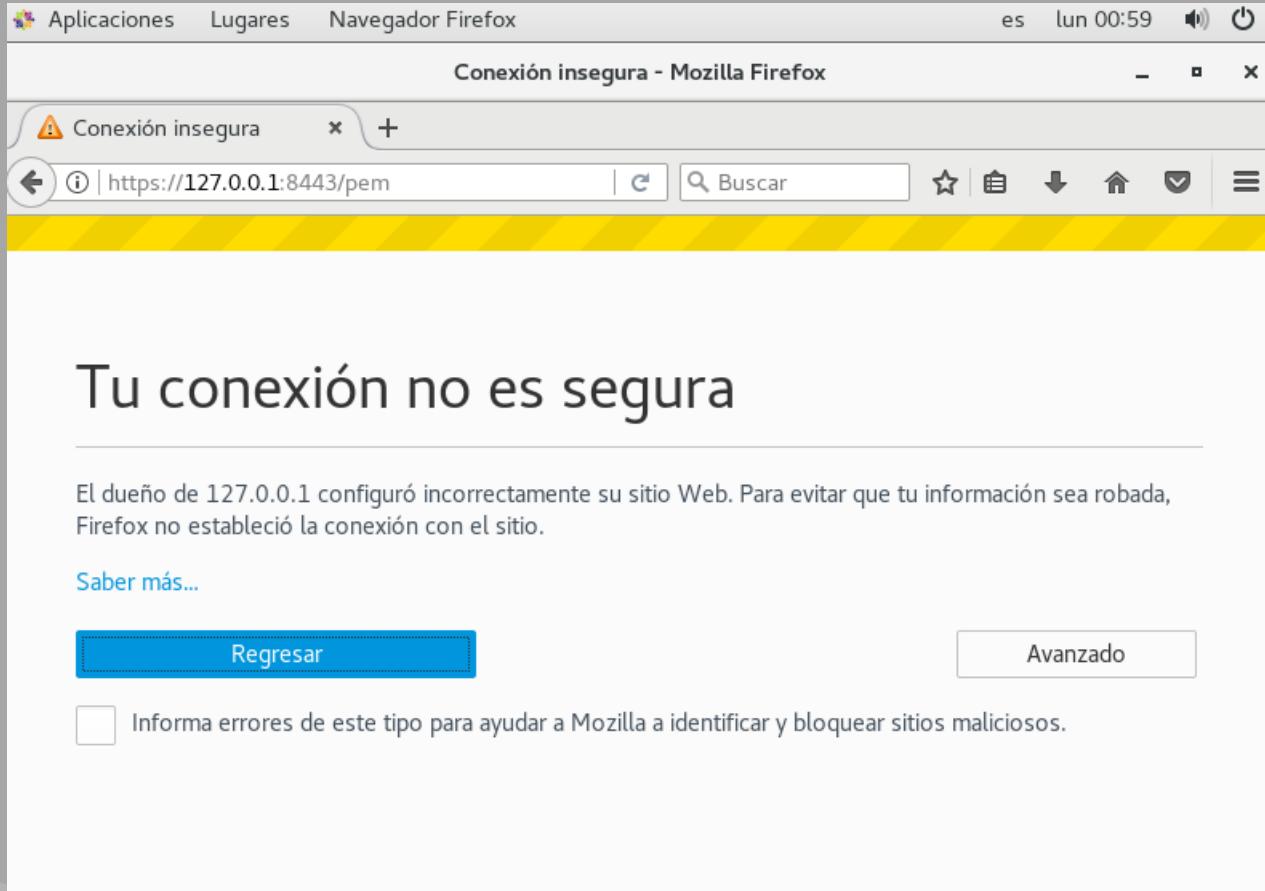
PEM ARCHITECTURE.



PEM ACCESO.



PEM ACCESO.



PEM ACCESO.



PEM ACCESO.

EDB-CENTOS7

Aplicaciones Lugares Navegador Firefox es lun 01:02

Postgres Enterprise Manager Login - Mozilla Firefox

Postgres Enterprise M... +

https://127.0.0.1:8443/pem/login/?next=%2F

Please log in to access this page.

 EDB

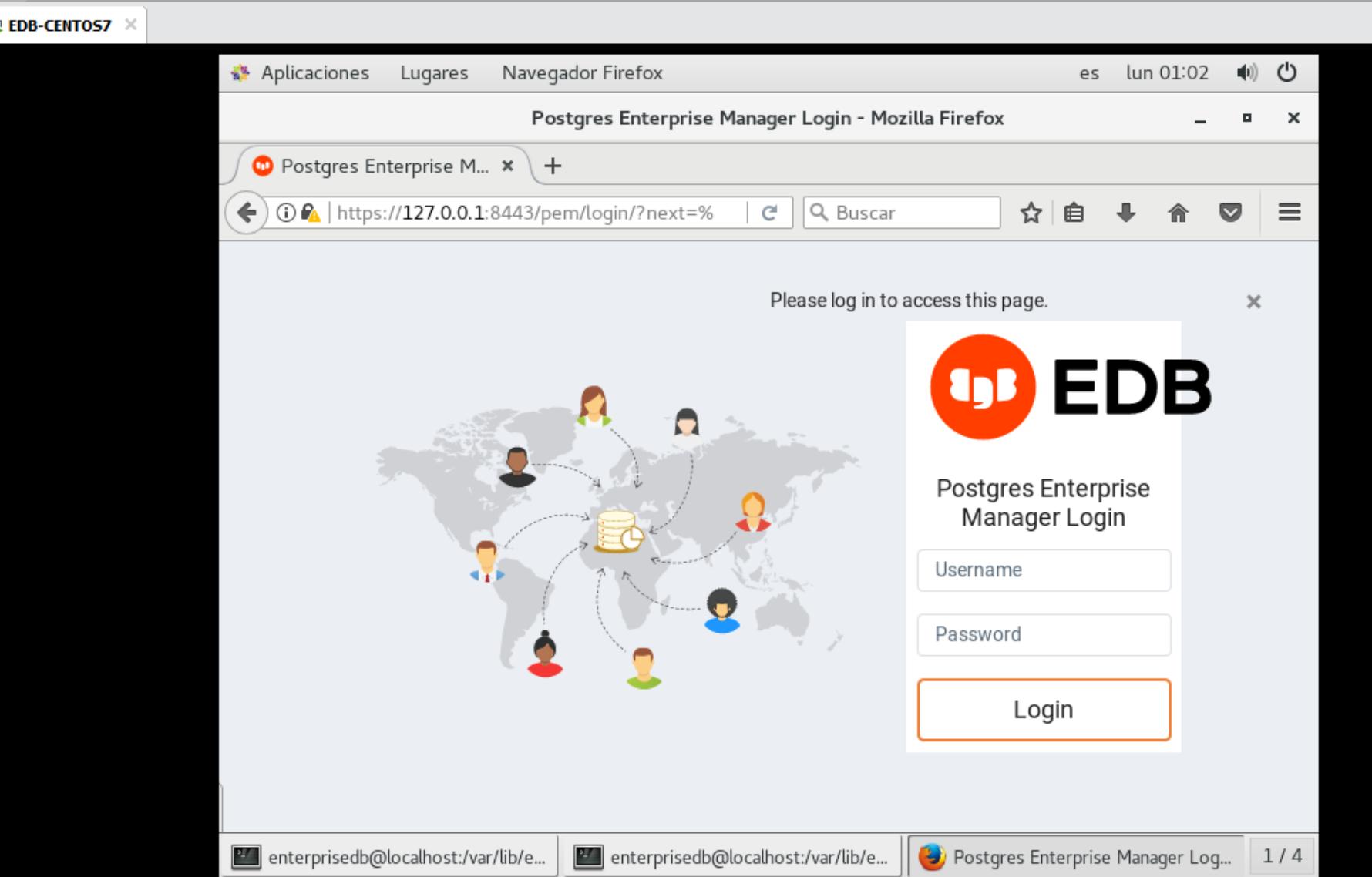
Postgres Enterprise Manager Login

Username

Password

Login

enterprise@localhost:/var/lib/e... enterprise@localhost:/var/lib/e... Postgres Enterprise Manager Log... 1 / 4



PEM GLOBAL DASHBOARD.

Aplicaciones Lugares Navegador Firefox es mar 22:40

Postgres Enterprise Manager - Mozilla Firefox

Postgres Enterprise M... x +

https://127.0.0.1:8443/pem/browser/ Buscar

File Object Management Dashboards Tools Help

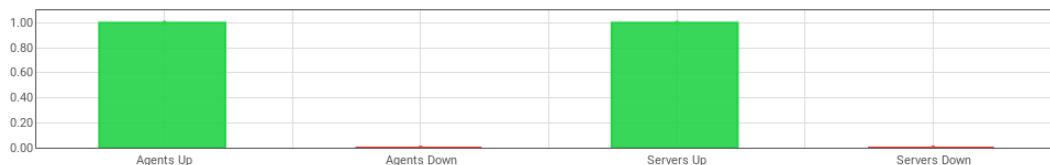
BROW Dashboard Properties SQL Statistics Dependencies Dependents Monitoring

Global Overview

Object Type System Status N/A Generated On 27/9/2022 22:40:29 No. of alerts 4 (Acknowledged: 0)

Enterprise Dashboard

Status



Agents Up Agents Down Servers Up Servers Down

Agent Status

Blackout	Status	Name	Alerts	Version	Processes	Threads	CPU Utilisation (%)	Memory Utilisation (%)	Swap Utilisation (%)	Disk Utilisation
<input type="checkbox"/>	 UP	Postgres Enterprise Manager Host	1	8.5.0	210	515	26.23	77.05	34.72	20.39

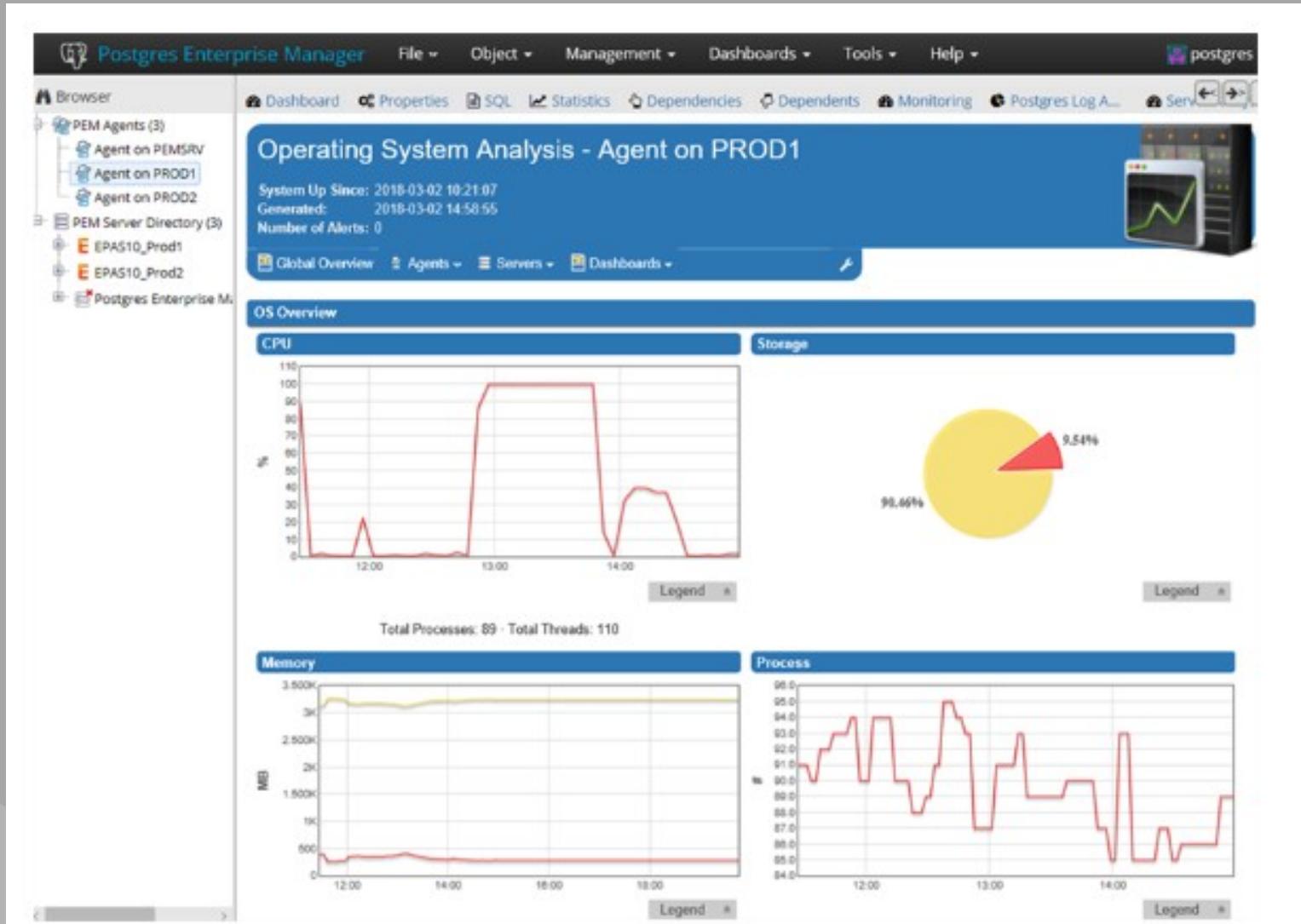
Server Status

Blackout	Status	Name	Connections	Alerts	Version	Remotely Monitored?
<input type="checkbox"/>	 UP	Postgres Enterprise Manager Server	12	3	PostgreSQL 13.8 (EnterpriseDB Advanced Server 13.8.12) on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44), 64-bit	No

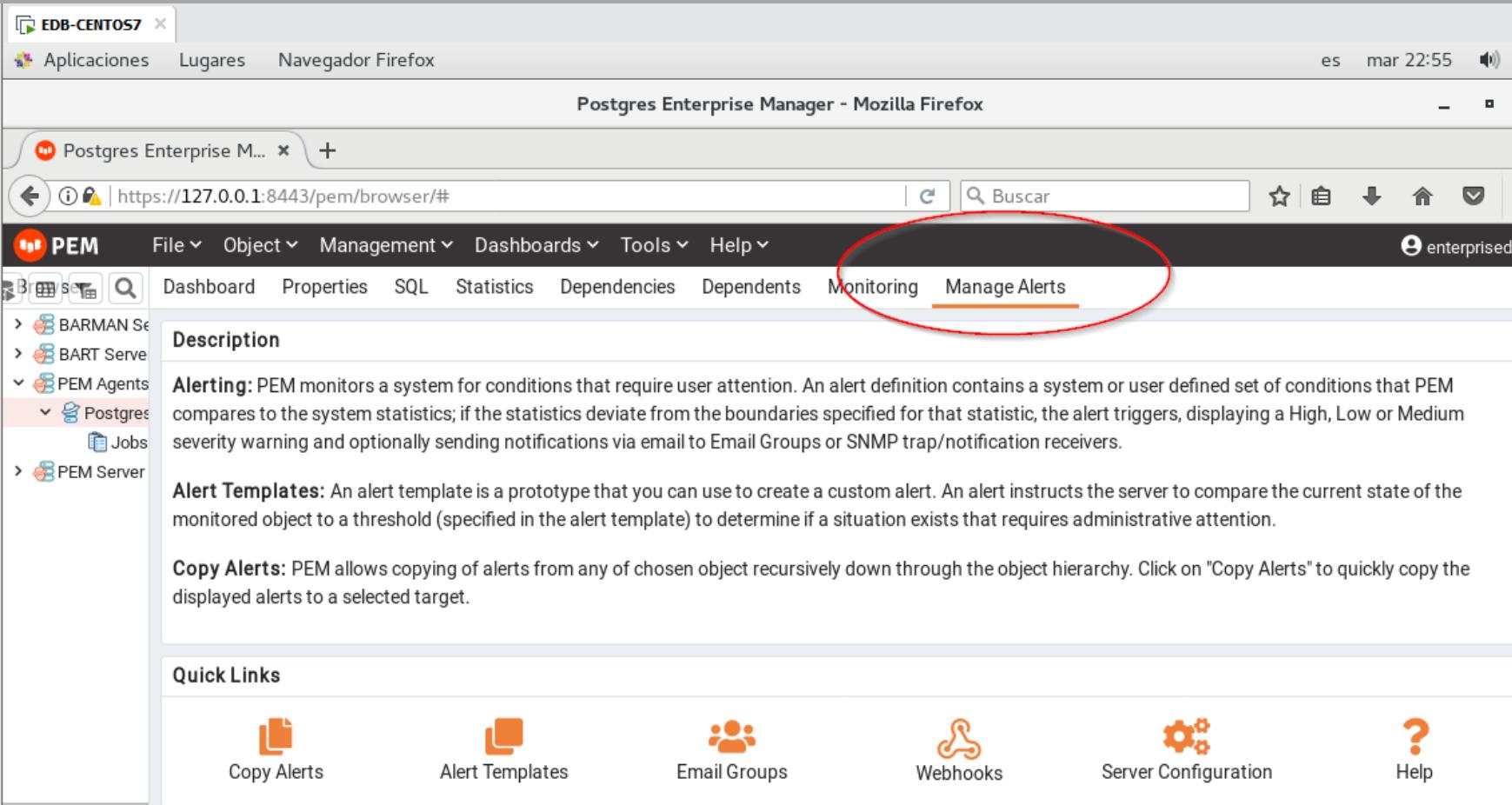
Alert Status

Alarm Type	Object Description	Alert Name	Value	Database	Schema	Package	Object	Alerting Since
▶ ● Low	Postgres Enterprise Manager Host	Swap consumption percentage	34.72%					2022-09-27 22:36:55
▶ ● Low	Postgres Enterprise Manager Server	Connections in idle state	7					2022-09-27 22:20:39
▶ ● High	Postgres Enterprise Manager Server	Last Vacuum	Never ran					2022-09-19 00:47:00

PEM AGENT OPERATING SYSTEM.



PEM AGENT | CREATING A CUSTOM ALERT.



The screenshot shows the PEM web interface running in Mozilla Firefox on a Linux system (EDB-CENTOS7). The browser title bar reads "Postgres Enterprise Manager - Mozilla Firefox". The main navigation bar includes links for File, Object, Management, Dashboards, Tools, and Help. A red oval highlights the "Manage Alerts" link in the top navigation bar. The left sidebar lists monitoring objects: BARMAN Server, BART Server, PEM Agents (with Postgres selected), Jobs, and PEM Server. The central content area is titled "Description" and contains three sections: Alerting, Alert Templates, and Copy Alerts. At the bottom, there is a "Quick Links" section with icons for Copy Alerts, Alert Templates, Email Groups, Webhooks, Server Configuration, and Help.

Description

Alerting: PEM monitors a system for conditions that require user attention. An alert definition contains a system or user defined set of conditions that PEM compares to the system statistics; if the statistics deviate from the boundaries specified for that statistic, the alert triggers, displaying a High, Low or Medium severity warning and optionally sending notifications via email to Email Groups or SNMP trap/notification receivers.

Alert Templates: An alert template is a prototype that you can use to create a custom alert. An alert instructs the server to compare the current state of the monitored object to a threshold (specified in the alert template) to determine if a situation exists that requires administrative attention.

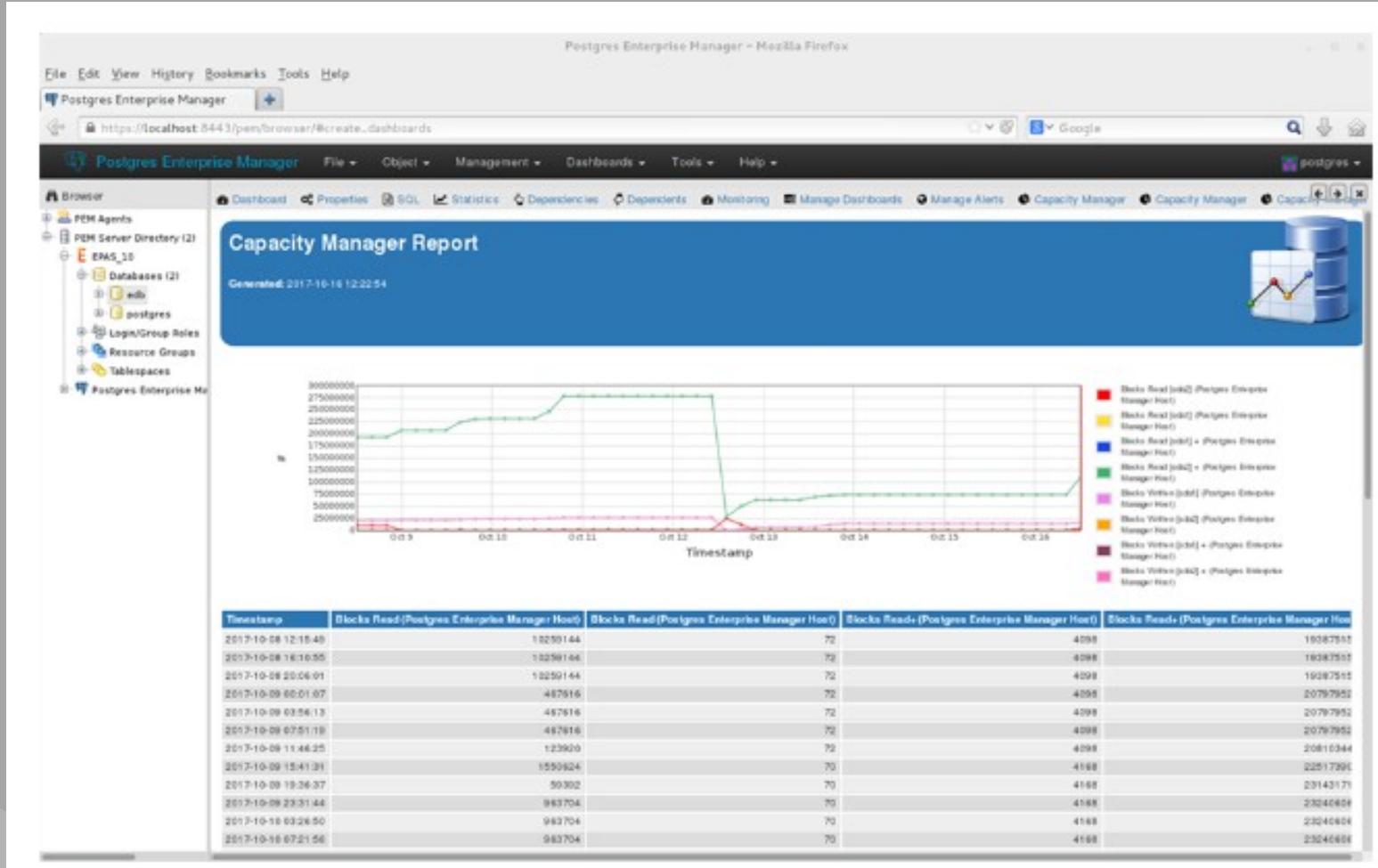
Copy Alerts: PEM allows copying of alerts from any of chosen object recursively down through the object hierarchy. Click on "Copy Alerts" to quickly copy the displayed alerts to a selected target.

Quick Links

- Copy Alerts
- Alert Templates
- Email Groups
- Webhooks
- Server Configuration
- Help

PEM CAPACITY MANAGER REPORT

Innovación tecnológica en defensa y seguridad

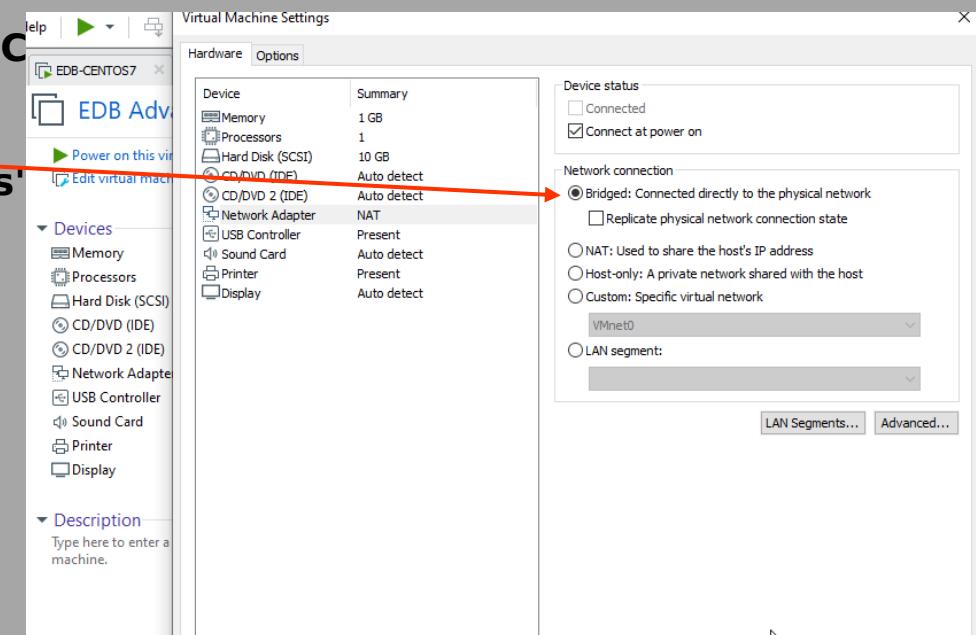


LABORATORIO 1.

Instalaremos lo necesario para poder realizar los laboratorios.

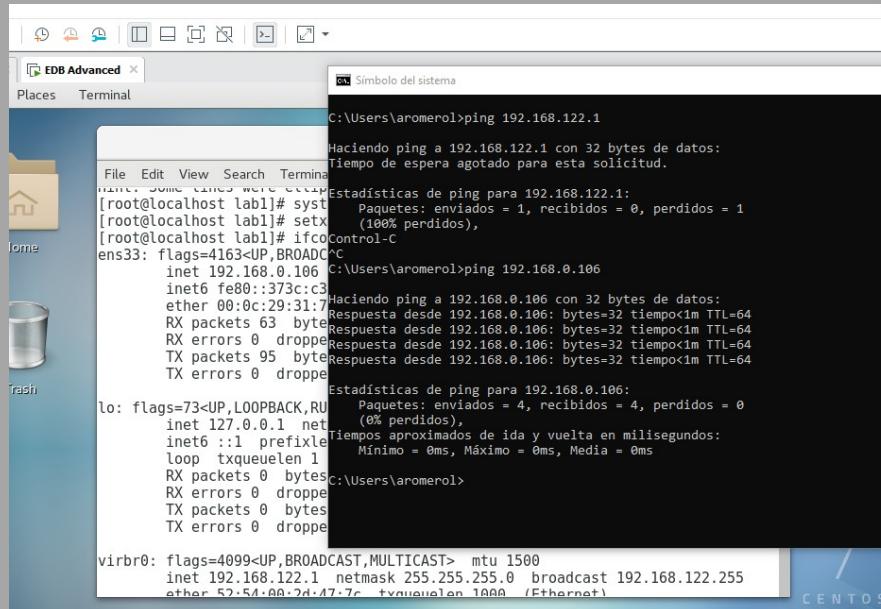
Prerequisitos:

- Una maquina virtual Centos 7.
- Usuario “root” Pass: 12345678.
- Usuario “lab1” Pass: 12345678.
- La máquina deberá tener acceso a internet , por eso tu tarjeta de red virtual tendrá que estar en “Nat” y tu laptop deberá tener conexión a Internet.
- Usuario enterprisedb que se creará más adelante deberá ser miembro de “sudoers” .
- Deberás de deshabilitar el Fire Wall de la máquina Centos 7.
○ [root@localhost etc]# systemctl stop firewalld
- Teclado en Español:
○ [root@localhost etc]# setxkbmap -layout 'es,es'
- Colocar tarjeta de red en Bridged.



LABORATORIO 1.

- Comprobar ping desde la PC hacia la máquina virtual.



```
C:\Users\aromerol>ping 192.168.122.1
Haciendo ping a 192.168.122.1 con 32 bytes de datos:
Tiempo de espera agotado para esta solicitud.

Estadísticas de ping para 192.168.122.1:
Paquetes: enviados = 1, recibidos = 0, perdidos = 1
(100% perdidos),
[root@localhost lab1]# setx
[root@localhost lab1]# ifconfig
[root@localhost lab1]# ifconfig
ens33: flags=4163<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.0.106  brd 192.168.0.255  netmask 255.255.255.0
              ether 52:54:00:7A:47:7c  txqueuelen 1000  (Ethernet)
          RX packets 63  bytes 6300
          RX errors 0  droppe
          TX packets 95  bytes 9500
          TX errors 0  droppe
lo: flags=73<UP,LOOPBACK,RUNNING  mtu 1643
        inet 127.0.0.1  netmask 255.0.0.0
              loop  txqueuelen 1
              RX packets 0  bytes 0
              RX errors 0  droppe
              TX packets 0  bytes 0
              TX errors 0  droppe
virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
              ether 52:54:00:7A:47:7c  txqueuelen 1000  (Ethernet)
```

LABORATORIO 1.

Comprobamos conexión a Internet:

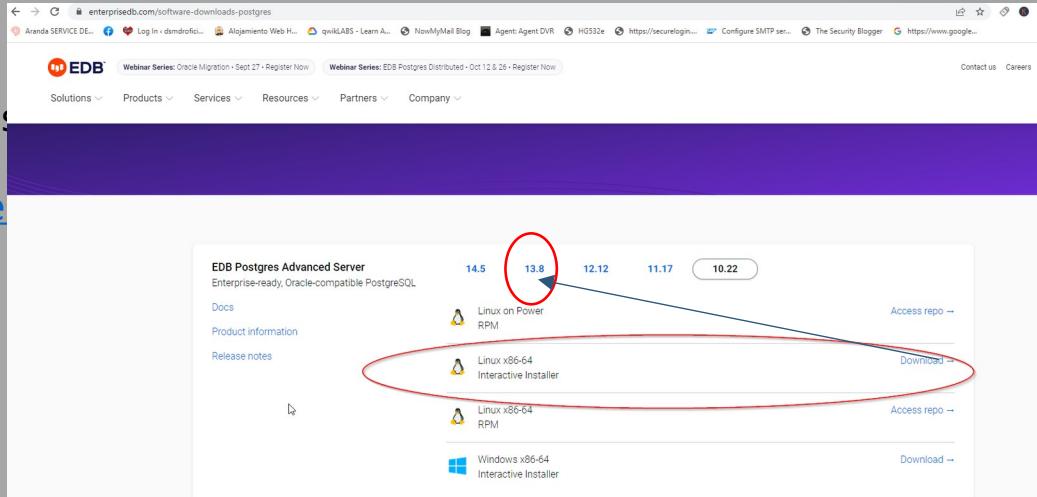
- wget google.com
- --2022-09-28 13:41:56-- <http://google.com/>
- Resolving google.com (google.com)... 142.250.65.110,
2607:f8b0:4012:818::200e
- Connecting to google.com (google.com)|142.250.65.110|:80... connected.
- HTTP request sent, awaiting response... 301 Moved Permanently
- Location: http://www.google.com/ [following]
- --2022-09-28 13:42:02-- <http://www.google.com/>
- Resolving www.google.com (www.google.com)... 142.251.34.36,
2607:f8b0:4012:814::2004
- Connecting to www.google.com (www.google.com)|142.251.34.36|:80...
connected.
- HTTP request sent, awaiting response... 200 OK
- Length: unspecified [text/html]
- Saving to: 'index.html'
- [<=>] 14,026 ---K/s in 0.008s
- 2022-09-28 13:42:02 (1.60 MB/s) - 'index.html' saved [14026]

LABORATORIO 1.

PASO 1

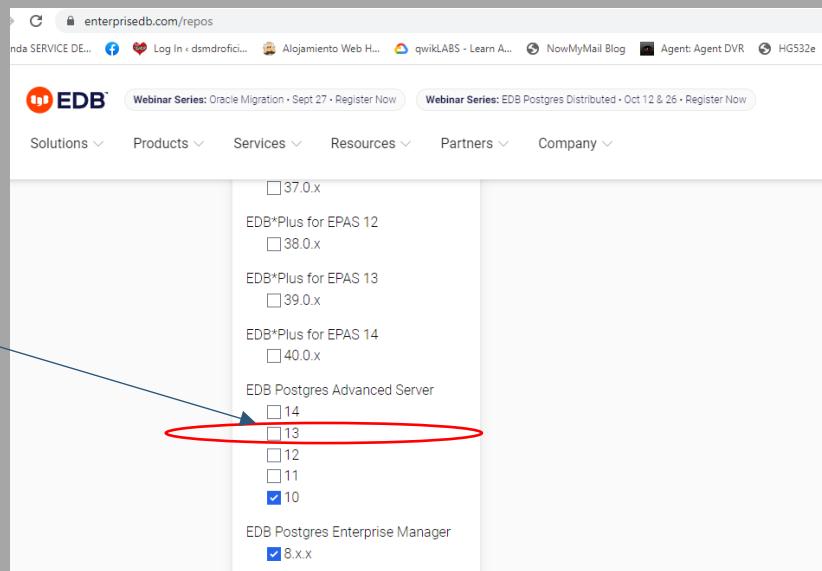
- Entramos a la siguiente liga:
- Debes de tener un usuario en la página de EDB , es decir que te dejen loguearte

<https://www.enterprisedb.com/software-downloads-postgres>



En la siguiente página elegimos:

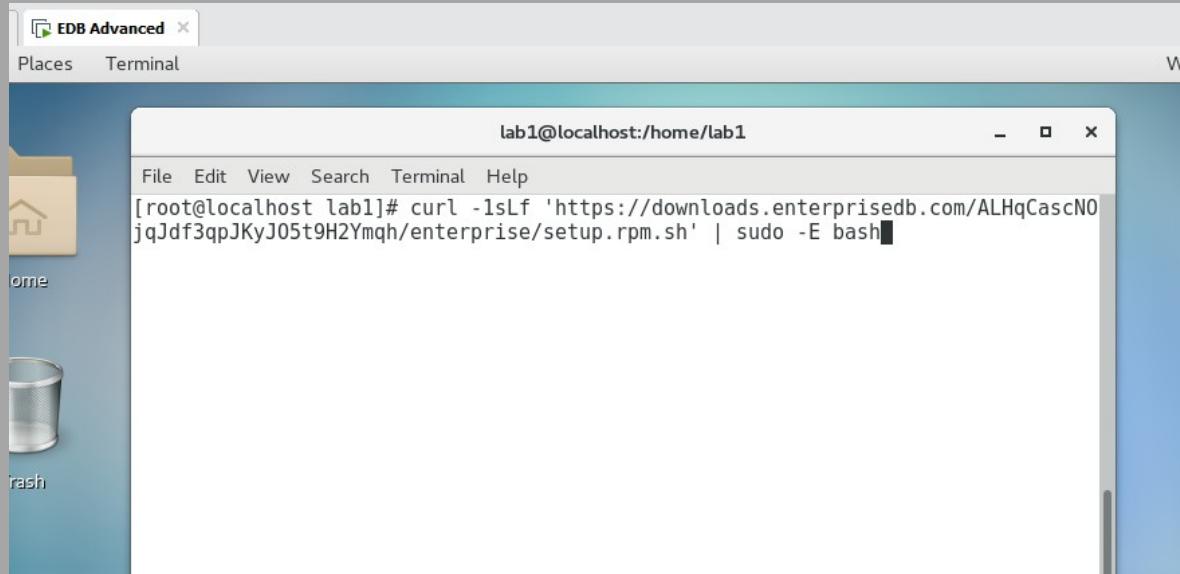
- EDB Postgres Advanced Server 13



LABORATORIO 1.

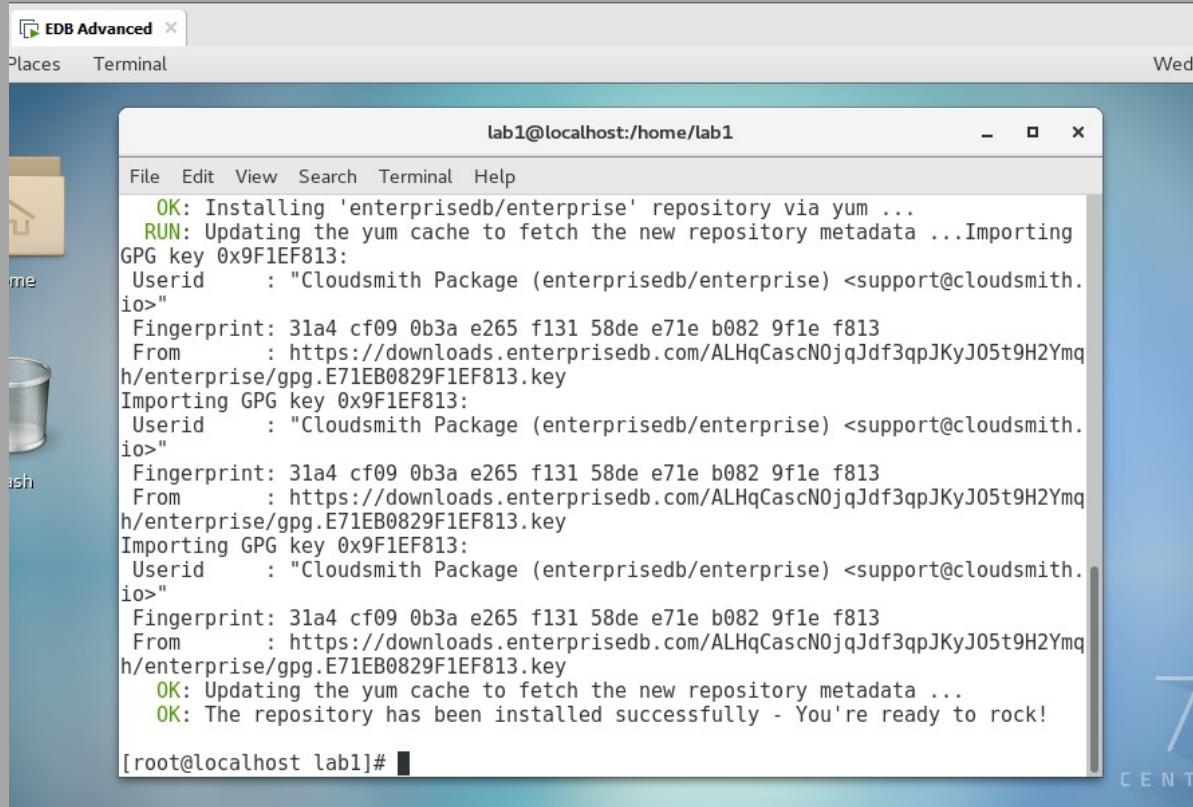
Copia este script:

- curl -sLf 'https://downloads.enterprisedb.com/ALHqCascNOjqJdf3qpJKyJO5t9H2Ymqh/enterprise/setup.rpm.sh' | sudo -E bash



LABORATORIO 1.

Comprobamos la ejecución del script.

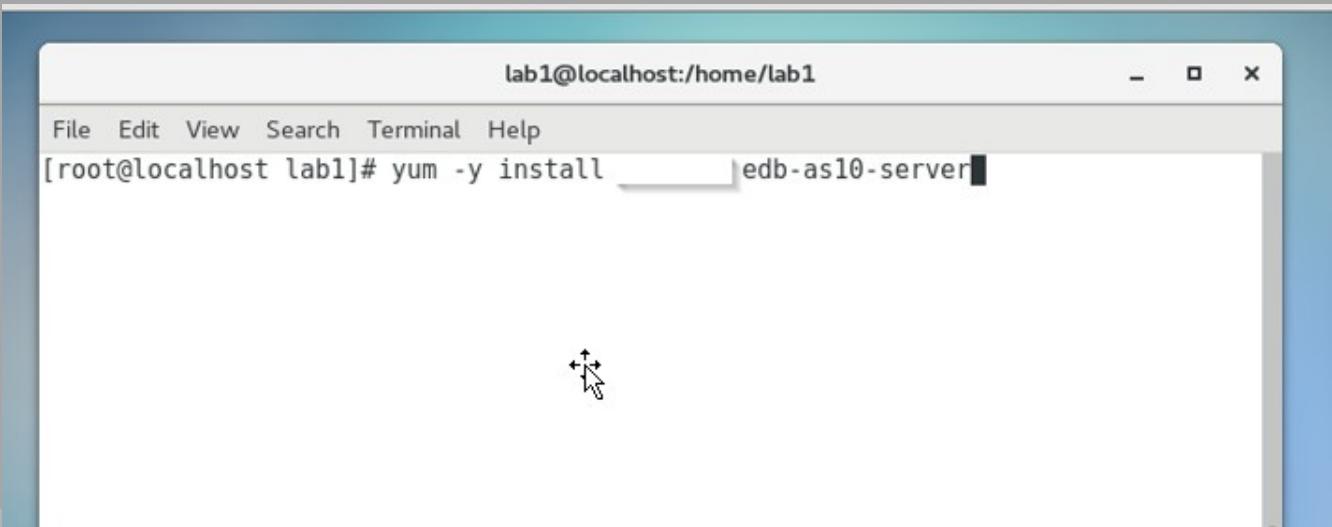


```
EDB Advanced x
Places Terminal
lab1@localhost:/home/lab1
File Edit View Search Terminal Help
OK: Installing 'enterprisedb/enterprise' repository via yum ...
RUN: Updating the yum cache to fetch the new repository metadata ...Importing
GPG key 0x9F1EF813:
Userid      : "Cloudsmith Package (enterprisedb/enterprise) <support@cloudsmith.
io>"
Fingerprint: 31a4 cf09 0b3a e265 f131 58de e71e b082 9f1e f813
From        : https://downloads.enterprisedb.com/ALHqCascN0jqJdf3qpJKyJ05t9H2Ymq
h/enterprise/gpg.E71EB0829F1EF813.key
Importing GPG key 0x9F1EF813:
Userid      : "Cloudsmith Package (enterprisedb/enterprise) <support@cloudsmith.
io>"
Fingerprint: 31a4 cf09 0b3a e265 f131 58de e71e b082 9f1e f813
From        : https://downloads.enterprisedb.com/ALHqCascN0jqJdf3qpJKyJ05t9H2Ymq
h/enterprise/gpg.E71EB0829F1EF813.key
Importing GPG key 0x9F1EF813:
Userid      : "Cloudsmith Package (enterprisedb/enterprise) <support@cloudsmith.
io>"
Fingerprint: 31a4 cf09 0b3a e265 f131 58de e71e b082 9f1e f813
From        : https://downloads.enterprisedb.com/ALHqCascN0jqJdf3qpJKyJ05t9H2Ymq
h/enterprise/gpg.E71EB0829F1EF813.key
OK: Updating the yum cache to fetch the new repository metadata ...
OK: The repository has been installed successfully - You're ready to rock!
[root@localhost lab1]#
```

LABORATORIO 1.

Paso 2.

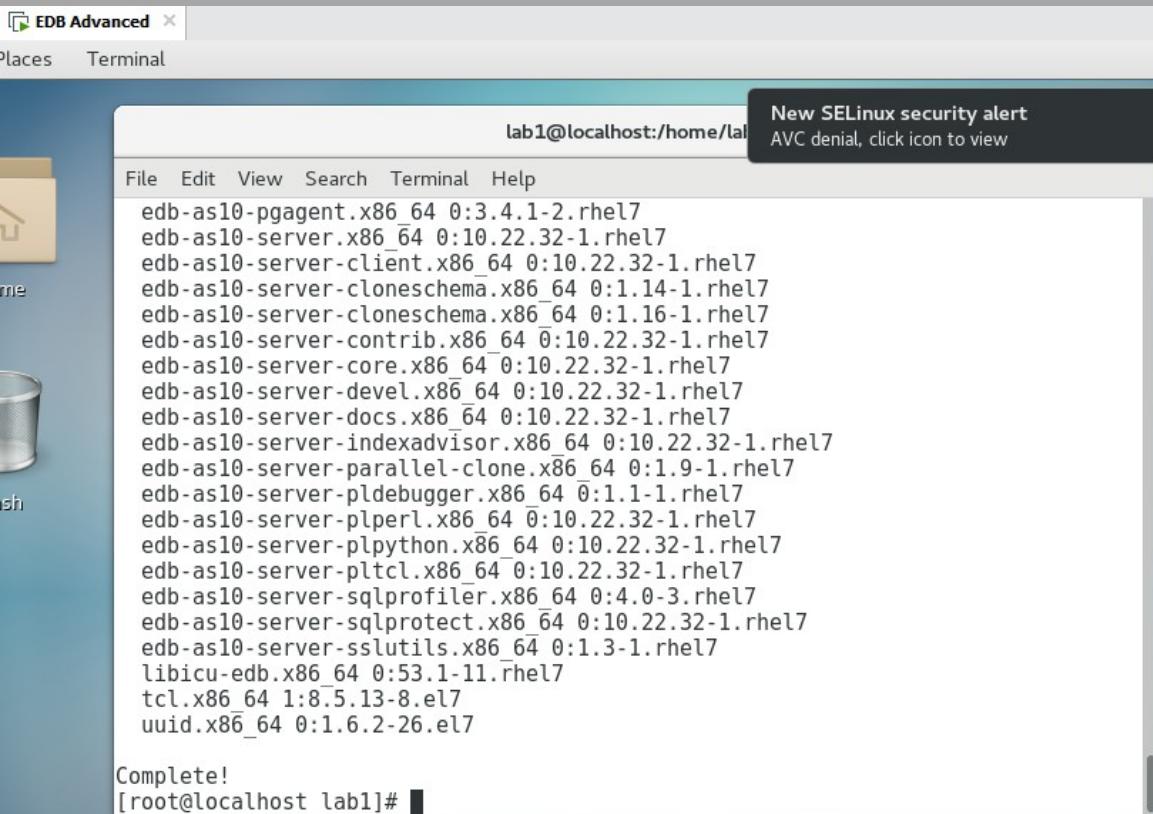
- Ejecutamos el siguiente script:
 - yum -y install edb-pem edb-as10-server
 -
 - root@localhost lab1]# yum -y install edb-as13-server --skip-broken



A screenshot of a terminal window titled "lab1@localhost:/home/lab1". The window has a standard Linux desktop interface with a title bar, menu bar, and scroll bars. The terminal prompt shows the user is root at localhost with the command "root@localhost lab1]#". The user is typing the command "yum -y install edb-as10-server" into the terminal. The cursor is positioned after the command line.

LABORATORIO 1.

Comprobamos instalación.



The screenshot shows a terminal window titled "EDB Advanced" running on a desktop environment. The window lists numerous packages installed on the system, including various EDB components like "edb-as10-pgagent", "edb-as10-server", and "edb-as10-server-client". At the top of the window, a message box displays a "New SELinux security alert" about an AVC denial, with instructions to click the icon to view it. The terminal command history at the bottom shows the user has run "ls" to list files and "Complete!" indicating the command has finished executing.

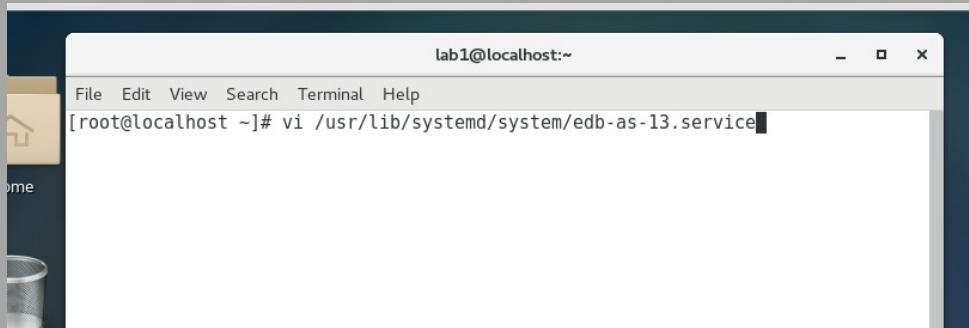
```
edb-as10-pgagent.x86_64 0:3.4.1-2.rhel7
edb-as10-server.x86_64 0:10.22.32-1.rhel7
edb-as10-server-client.x86_64 0:10.22.32-1.rhel7
edb-as10-server-cloneschema.x86_64 0:1.14-1.rhel7
edb-as10-server-cloneschema.x86_64 0:1.16-1.rhel7
edb-as10-server-contrib.x86_64 0:10.22.32-1.rhel7
edb-as10-server-core.x86_64 0:10.22.32-1.rhel7
edb-as10-server-devel.x86_64 0:10.22.32-1.rhel7
edb-as10-server-docs.x86_64 0:10.22.32-1.rhel7
edb-as10-server-indexadvisor.x86_64 0:10.22.32-1.rhel7
edb-as10-server-parallel-clone.x86_64 0:1.9-1.rhel7
edb-as10-server-pldebugger.x86_64 0:1.1-1.rhel7
edb-as10-server-plperl.x86_64 0:10.22.32-1.rhel7
edb-as10-server-plpython.x86_64 0:10.22.32-1.rhel7
edb-as10-server-pltcl.x86_64 0:10.22.32-1.rhel7
edb-as10-server-sqlprofiler.x86_64 0:4.0-3.rhel7
edb-as10-server-sqlprotect.x86_64 0:10.22.32-1.rhel7
edb-as10-server-sslutils.x86_64 0:1.3-1.rhel7
libicu-edb.x86_64 0:53.1-11.rhel7
tcl.x86_64 1:8.5.13-8.el7
uuid.x86_64 0:1.6.2-26.el7

Complete!
[root@localhost lab1]#
```

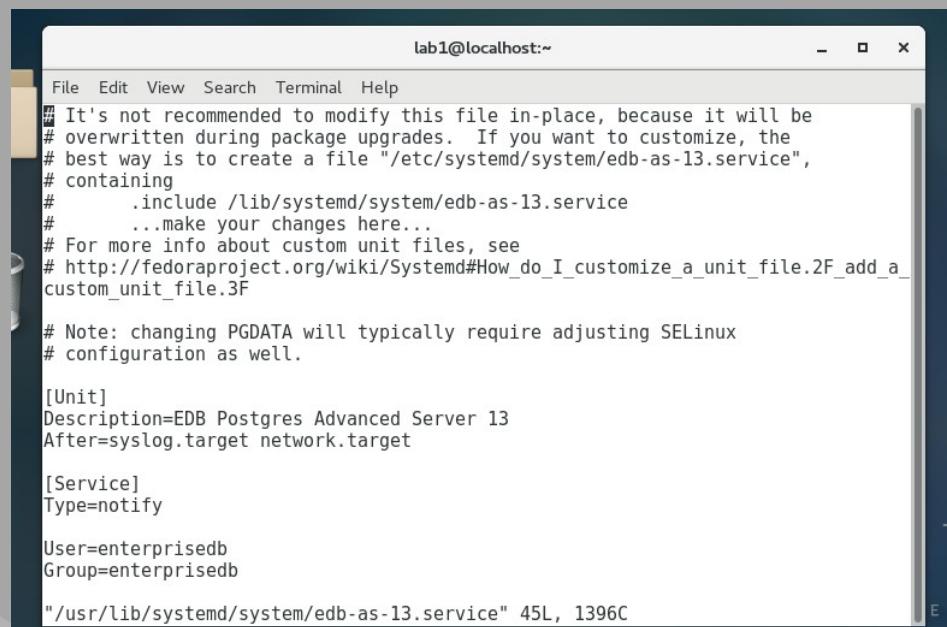
LABORATORIO 1.

Instalación

- Verificamos el siguiente (No hacer nada) solo revisar el archivo:



```
lab1@localhost:~  
File Edit View Search Terminal Help  
[root@localhost ~]# vi /usr/lib/systemd/system/edb-as-13.service
```

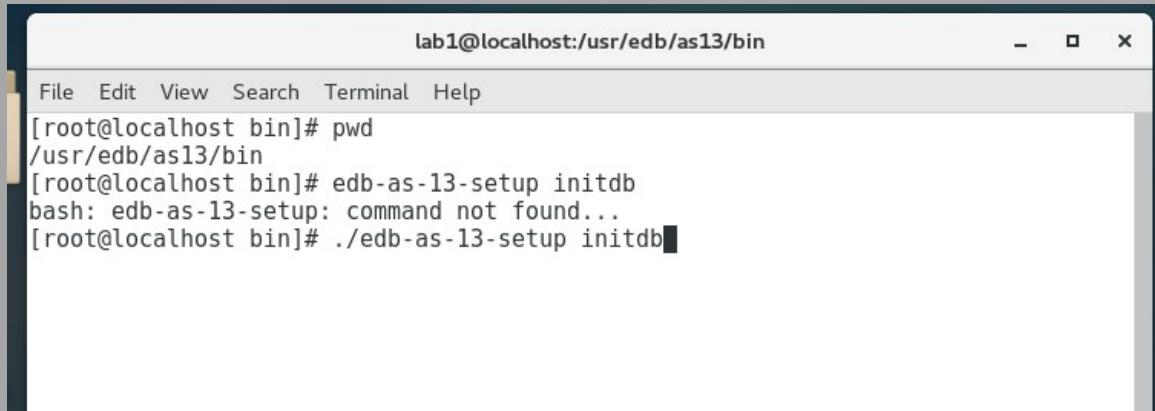


```
lab1@localhost:~  
File Edit View Search Terminal Help  
# It's not recommended to modify this file in-place, because it will be  
# overwritten during package upgrades. If you want to customize, the  
# best way is to create a file "/etc/systemd/system/edb-as-13.service",  
# containing  
#     .include /lib/systemd/system/edb-as-13.service  
#     ...make your changes here...  
# For more info about custom unit files, see  
# http://fedoraproject.org/wiki/Systemd#How_do_I_customize_a_unit_file.2F_add_a_  
# custom_unit_file.3F  
# Note: changing PGDATA will typically require adjusting SELinux  
# configuration as well.  
  
[Unit]  
Description=EDB Postgres Advanced Server 13  
After=syslog.target network.target  
  
[Service]  
Type=notify  
  
User=enterprisedb  
Group=enterprisedb  
  
"/usr/lib/systemd/system/edb-as-13.service" 45L, 1396C
```

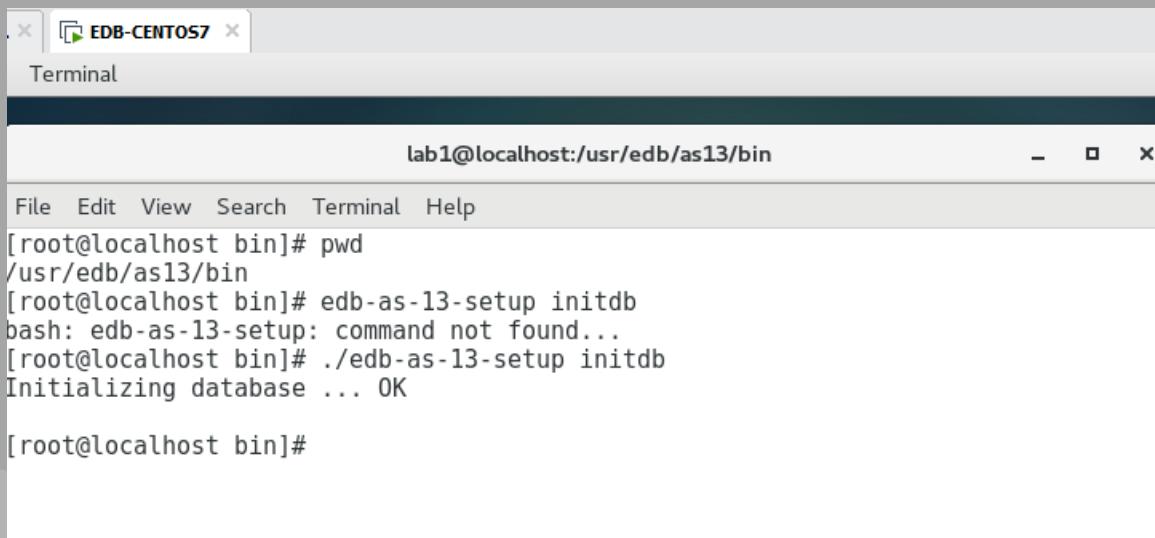
LABORATORIO 1.

Instalación

- Entramos al siguiente directorio:



```
lab1@localhost:/usr/edb/as13/bin
File Edit View Search Terminal Help
[root@localhost bin]# pwd
/usr/edb/as13/bin
[root@localhost bin]# edb-as-13-setup initdb
bash: edb-as-13-setup: command not found...
[root@localhost bin]# ./edb-as-13-setup initdb
```



EDB-CENTOS7 Terminal

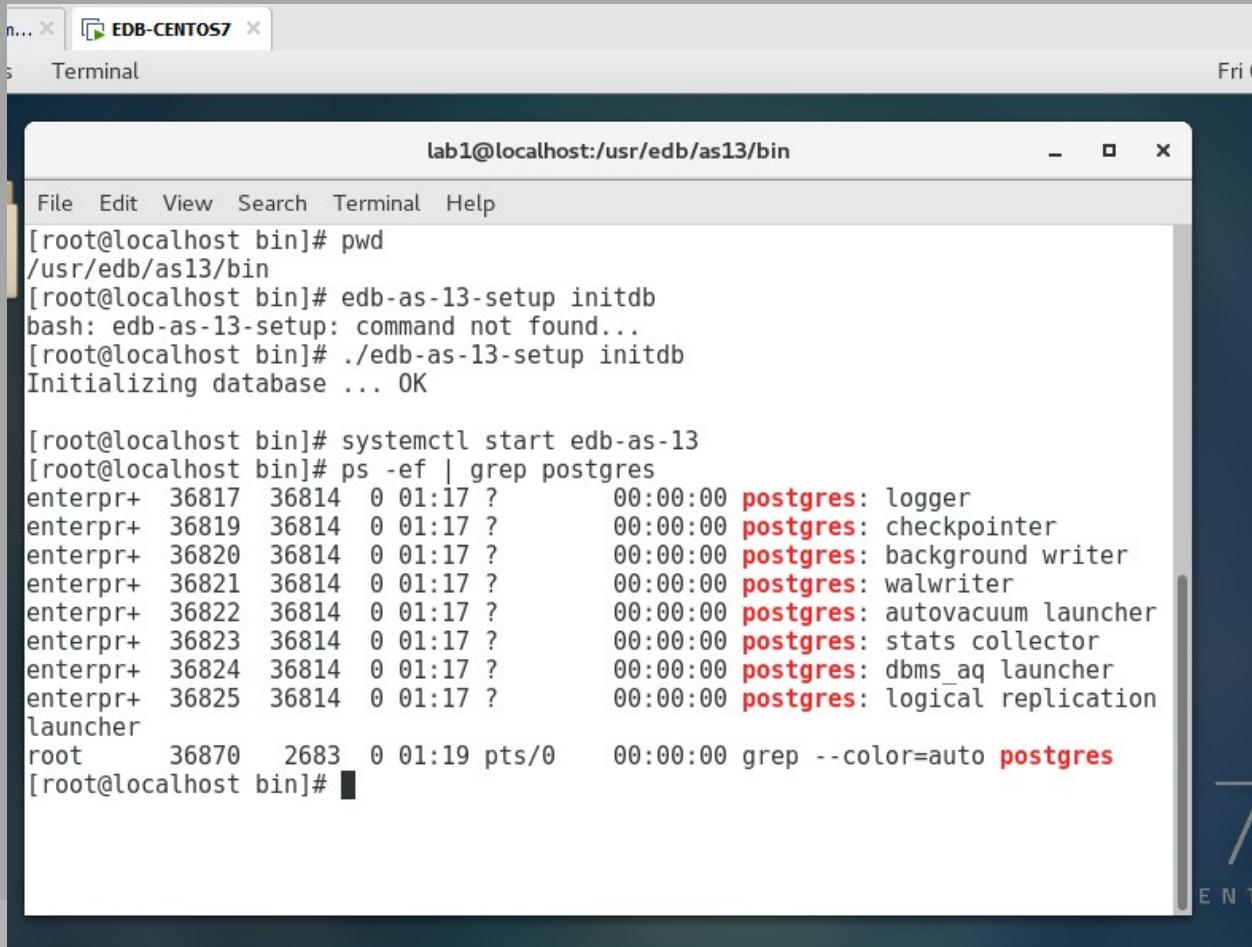
```
lab1@localhost:/usr/edb/as13/bin
File Edit View Search Terminal Help
[root@localhost bin]# pwd
/usr/edb/as13/bin
[root@localhost bin]# edb-as-13-setup initdb
bash: edb-as-13-setup: command not found...
[root@localhost bin]# ./edb-as-13-setup initdb
Initializing database ... OK

[root@localhost bin]#
```

LABORATORIO 1.

Instalación

- Siguiente paso verificamos que nuestra base de datos este funcionando:



The screenshot shows a terminal window titled "Terminal" running on a system named "EDB-CENTOS7". The window title bar also displays "Terminal". The terminal session is as follows:

```
lab1@localhost:/usr/edb/as13/bin
File Edit View Search Terminal Help
[root@localhost bin]# pwd
/usr/edb/as13/bin
[root@localhost bin]# edb-as-13-setup initdb
bash: edb-as-13-setup: command not found...
[root@localhost bin]# ./edb-as-13-setup initdb
Initializing database ... OK

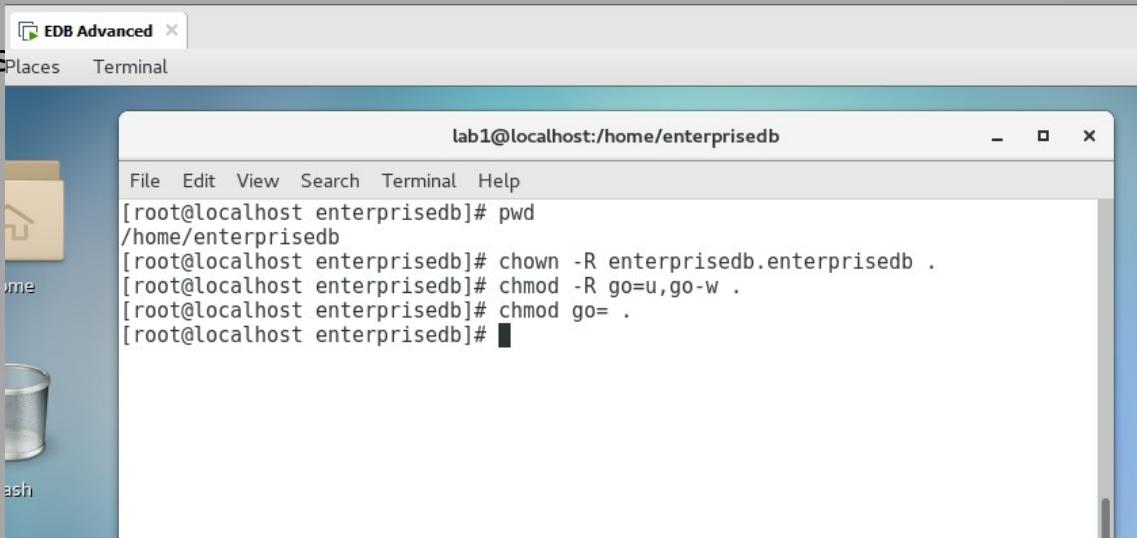
[root@localhost bin]# systemctl start edb-as-13
[root@localhost bin]# ps -ef | grep postgres
enterpr+ 36817 36814 0 01:17 ? 00:00:00 postgres: logger
enterpr+ 36819 36814 0 01:17 ? 00:00:00 postgres: checkpointer
enterpr+ 36820 36814 0 01:17 ? 00:00:00 postgres: background writer
enterpr+ 36821 36814 0 01:17 ? 00:00:00 postgres: walwriter
enterpr+ 36822 36814 0 01:17 ? 00:00:00 postgres: autovacuum launcher
enterpr+ 36823 36814 0 01:17 ? 00:00:00 postgres: stats collector
enterpr+ 36824 36814 0 01:17 ? 00:00:00 postgres: dbms_aq launcher
enterpr+ 36825 36814 0 01:17 ? 00:00:00 postgres: logical replication
launcher
root 36870 2683 0 01:19 pts/0 00:00:00 grep --color=auto postgres
[root@localhost bin]#
```

The terminal window has a dark blue header and a light gray body. The text is white on a black background. The number "7" is visible in the bottom right corner of the terminal window.

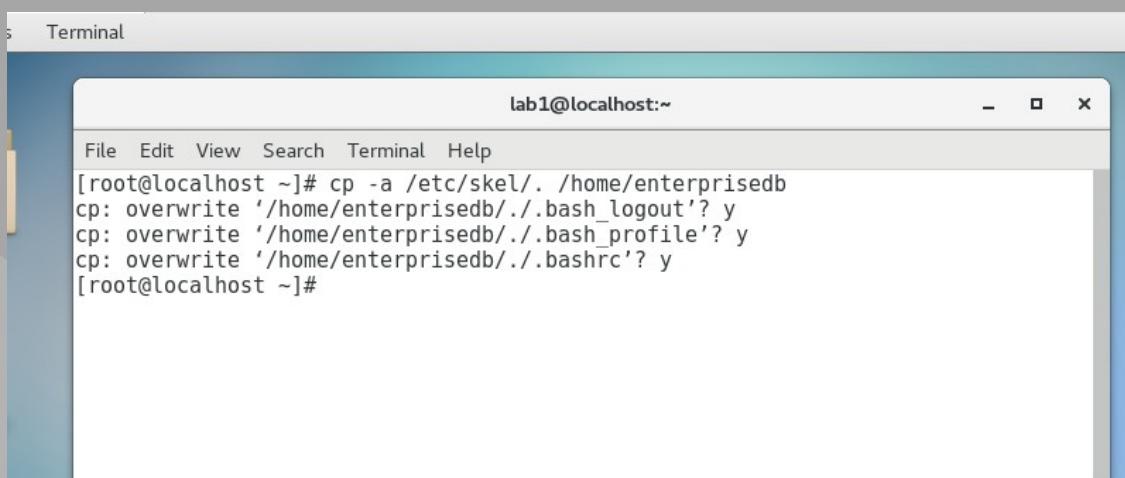
LABORATORIO 1.

Instalación

- Configuración de Home del usuario enterprise



```
lab1@localhost:/home/enterprisedb
File Edit View Search Terminal Help
[root@localhost enterprisedb]# pwd
/home/enterprisedb
[root@localhost enterprisedb]# chown -R enterprisedb.enterprisedb .
[root@localhost enterprisedb]# chmod -R go=u,go-w .
[root@localhost enterprisedb]# chmod go= .
[root@localhost enterprisedb]#
```

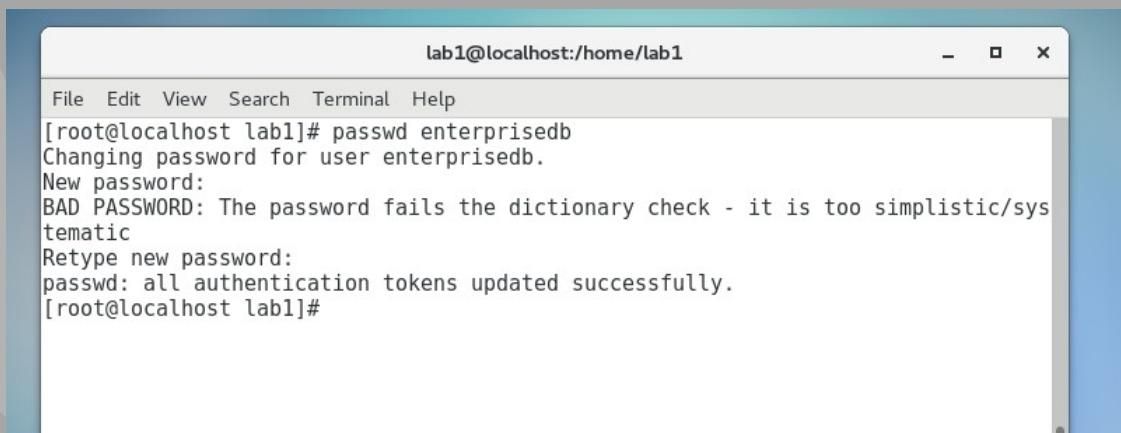
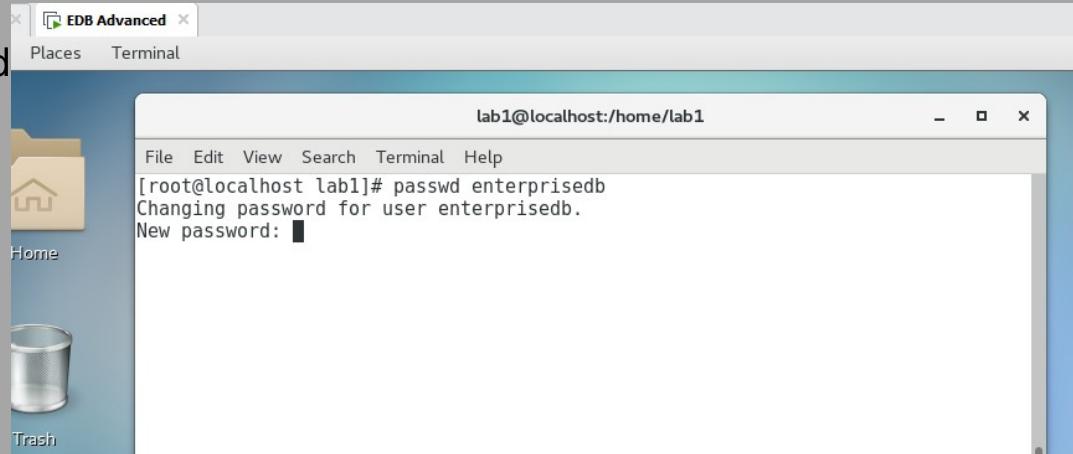


```
Terminal
lab1@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# cp -a /etc/skel/. /home/enterprisedb
cp: overwrite '/home/enterprisedb/.bash_logout'? y
cp: overwrite '/home/enterprisedb/.bash_profile'? y
cp: overwrite '/home/enterprisedb/.bashrc'? y
[root@localhost ~]#
```

LABORATORIO 1.

Instalación

- Configuración de Home del usuario enterprise
- Pass: 12345678



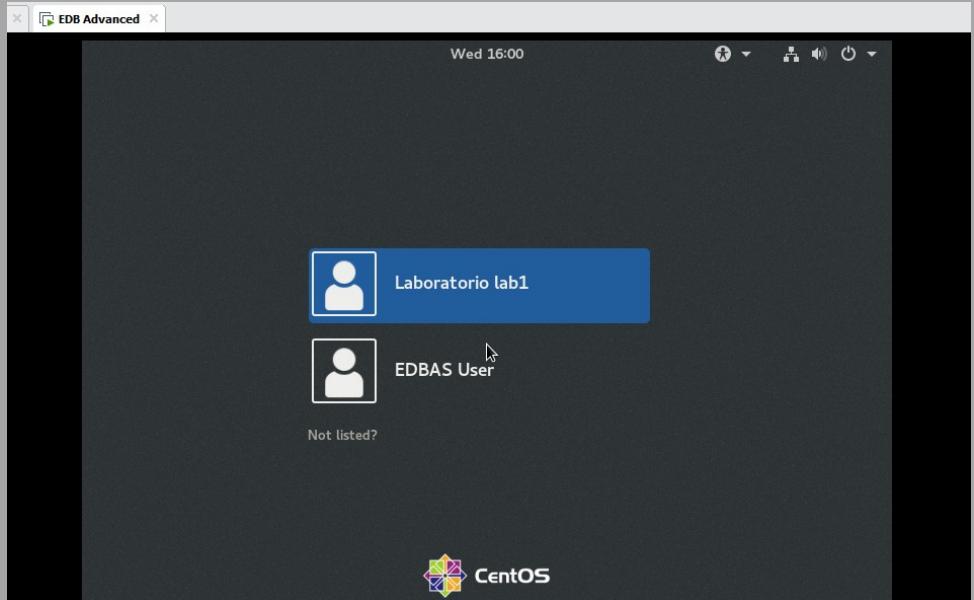
```
lab1@localhost:/home/lab1
File Edit View Search Terminal Help
[root@localhost lab1]# passwd enterprise
Changing password for user enterprise.
New password:
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost lab1]#
```

- Debemos reiniciar la maquina virtual.

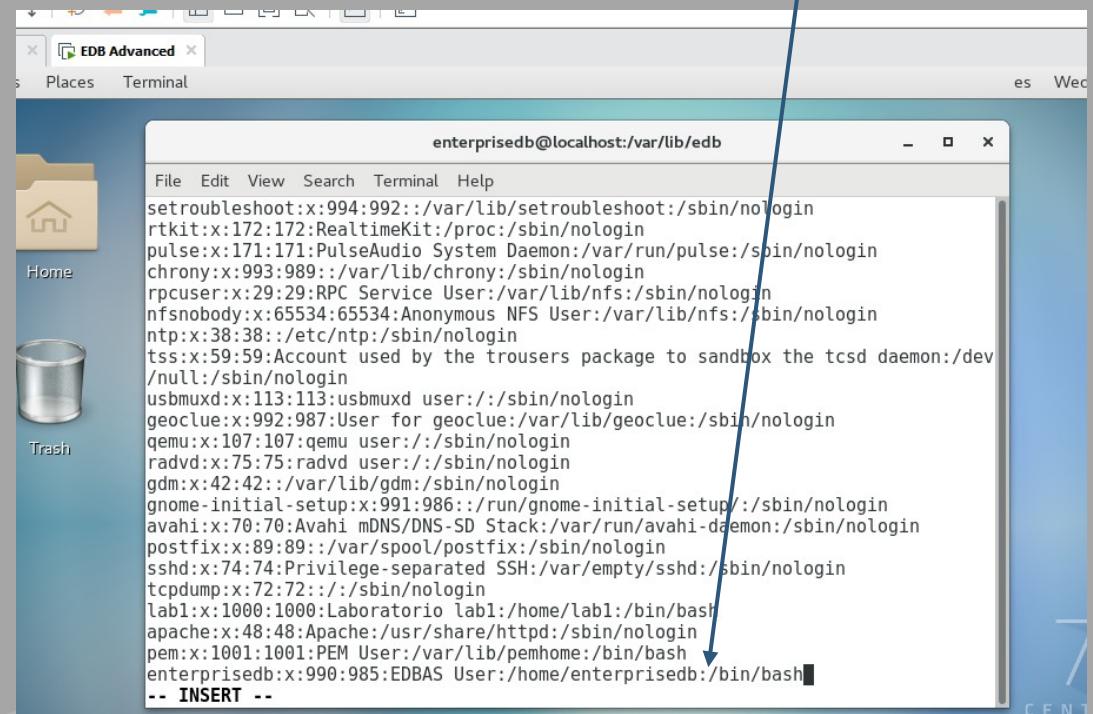
LABORATORIO 1.

Instalación

Configuración de Home del usuario enterpriseDB.



- Configuración de /etc/passwd

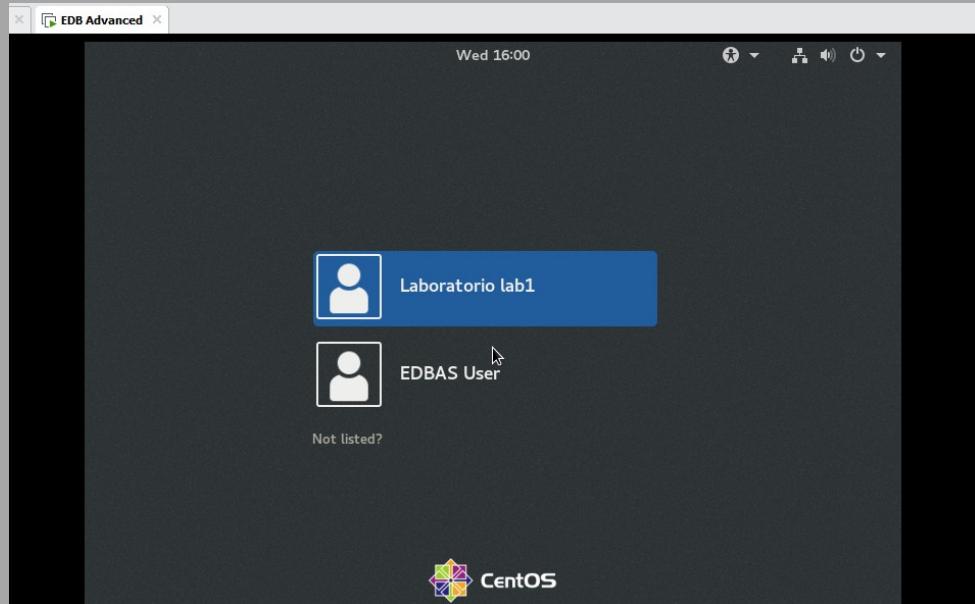


```
enterprisedb@localhost:/var/lib/edb
File Edit View Search Terminal Help
setroubleshoot:x:994:992::/var/lib/setroubleshoot:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc/sbin/nologin
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
chrony:x:993:989::/var/lib/chrony:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/sbin/nologin
geoclue:x:992:987:User for geoclue:/var/lib/geoclue:/sbin/nologin
qemu:x:107:107:qemu user:/sbin/nologin
radvd:x:75:75:radvd user:/sbin/nologin
gdm:x:42:42:/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:991:986:/run/gnome-initial-setup:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72::/sbin/nologin
lab1:x:1000:1000:Laboratorio lab1:/home/lab1:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
pem:x:1001:1001:PEM User:/var/lib/pemhome:/bin/bash
enterprisedb:x:990:985:EDBAS User:/home/enterprisedb:/bin/bash
-- INSERT --
```

LABORATORIO 1.

Instalación

Entramos de nuevo con user: lab1 pass: 12345678

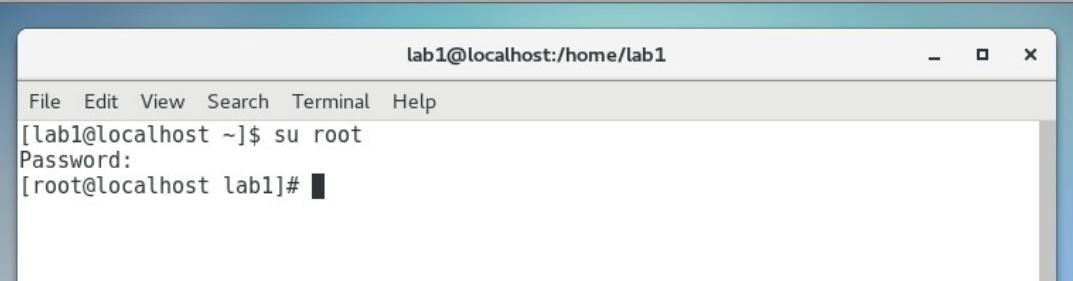


```
lab1@localhost:/home/lab1
File Edit View Search Terminal Help
[lab1@localhost ~]$ su root
Password:
[root@localhost lab1]#
```

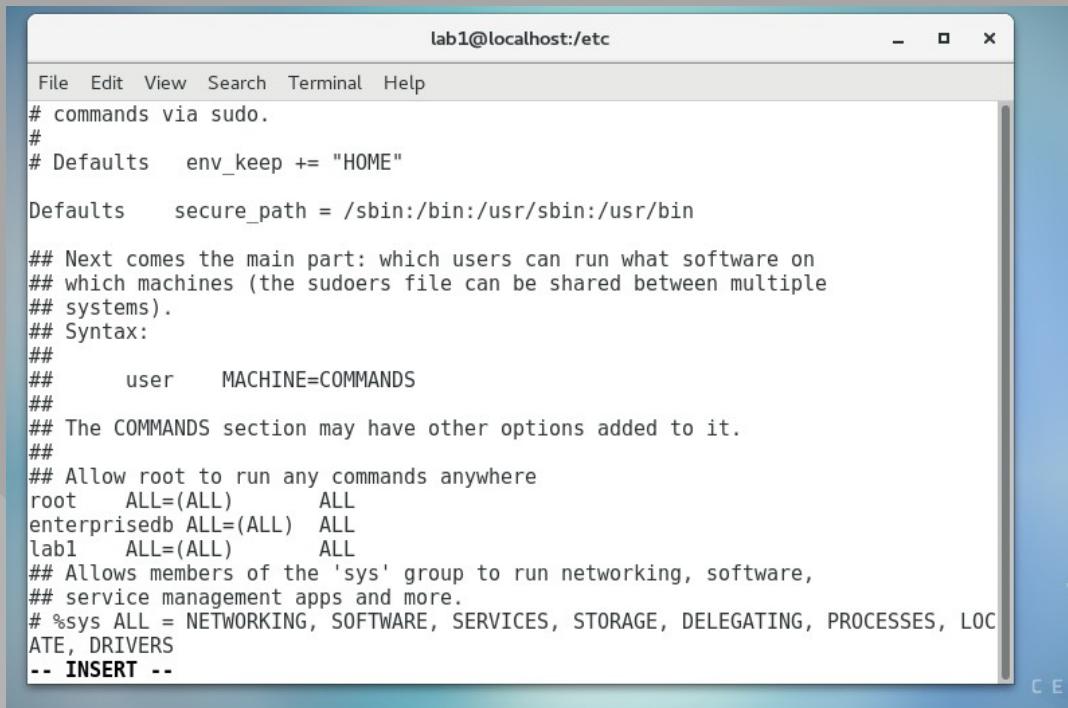
LABORATORIO 1.

Instalación

- Entramos de nuevo con user: lab1 pass: 12345
- Cd /etc.
- Vi sudoers.



```
lab1@localhost:/home/lab1
File Edit View Search Terminal Help
[lab1@localhost ~]$ su root
Password:
[root@localhost lab1]#
```



```
lab1@localhost:/etc
File Edit View Search Terminal Help
# commands via sudo.
#
# Defaults env_keep += "HOME"

Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##     user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)      ALL
enterprisedb ALL=(ALL)      ALL
lab1    ALL=(ALL)      ALL
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOC
ATE, DRIVERS
-- INSERT --
```

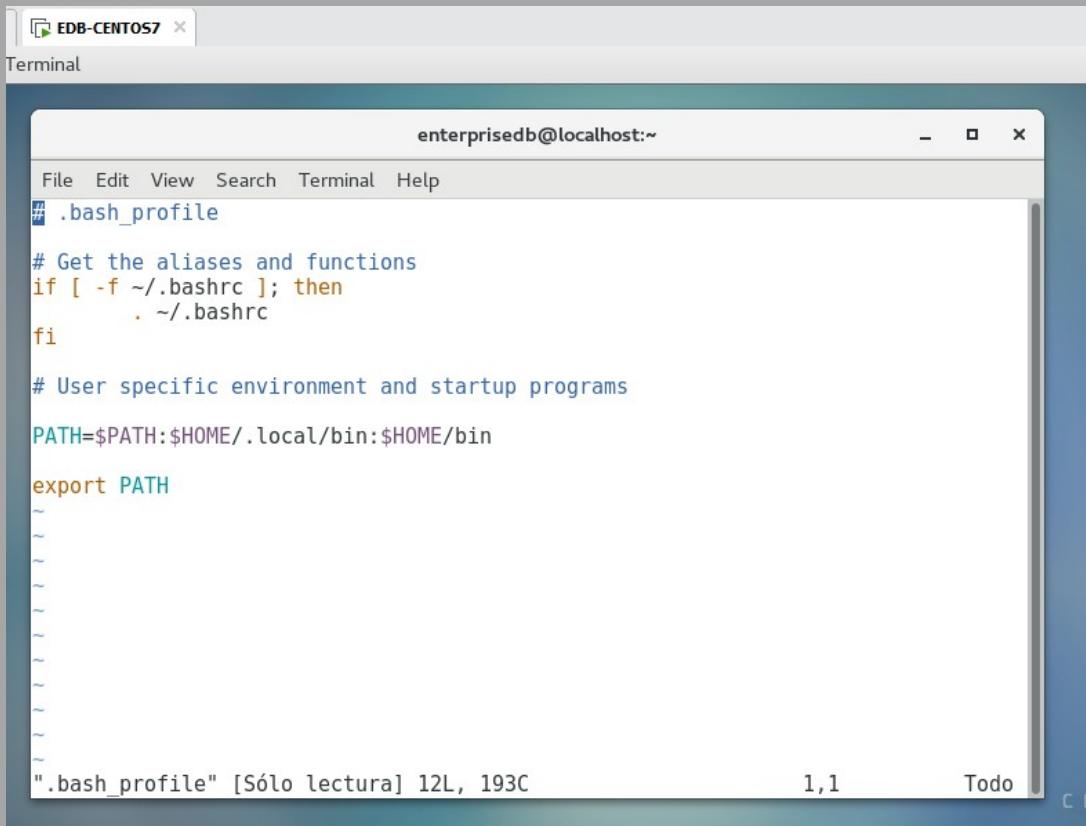
LABORATORIO 1.

Instalación

Configuración de Variables del usuario enterpriseDB (Este usuario lo crea automáticamente)

Ejecutamos: su enterpriseDB

vi .bash_profile



```
edb-centos7 x | Terminal
enterprisedb@localhost:~
```

```
.bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export PATH
```

".bash_profile" [Sólo lectura] 12L, 193C 1,1 Todo C E

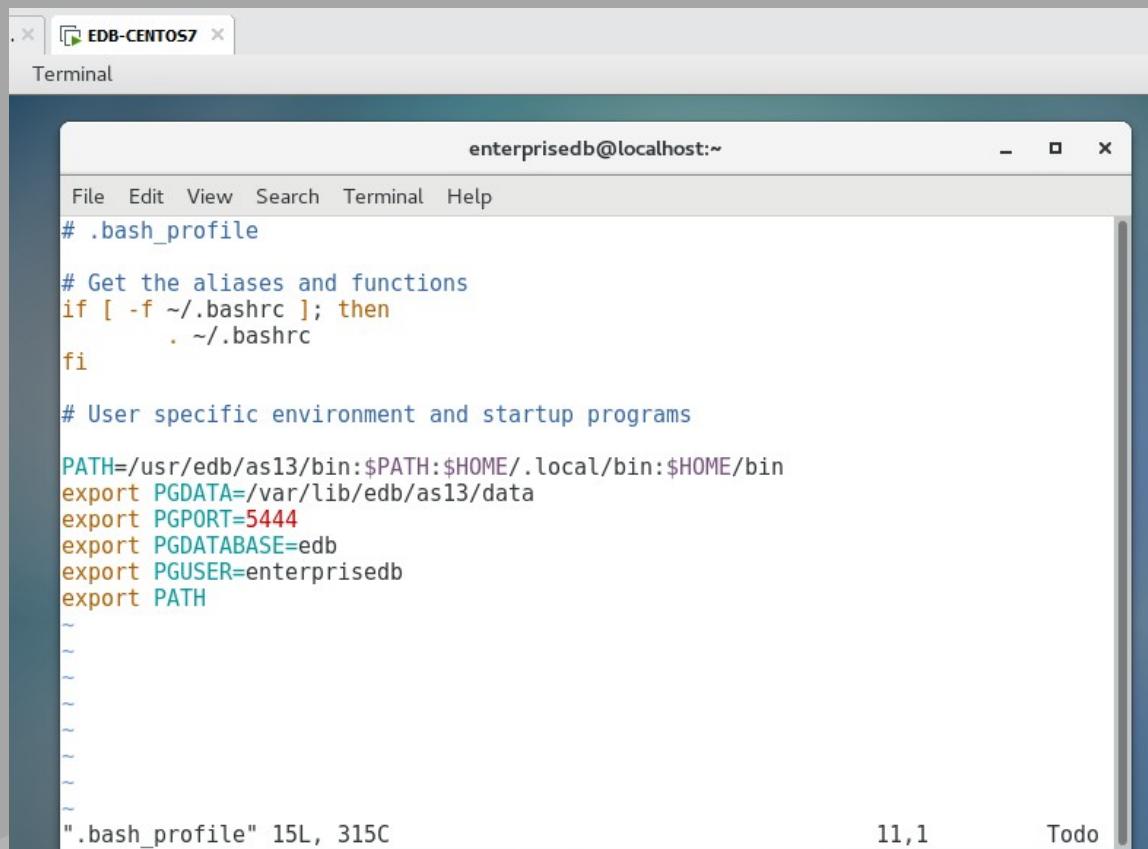
LABORATORIO 1.

Instalación

Configuración de Variables del usuario enterpriseDB

- Insertamos lo siguiente:

- `path=/usr/edb/as13/bin: $PATH: $HOME/.local/bin: $HOME/bin`
- `export PGDATA=/var/lib/edb/as13/data`
- `export PGPORT=5444`
- `lexport PGDATABASE=edb`
- `export PGUSER=enterpriseDB`
- `export PATH`



```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=/usr/edb/as13/bin:$PATH:$HOME/.local/bin:$HOME/bin
export PGDATA=/var/lib/edb/as13/data
export PGPORT=5444
export PGDATABASE=edb
export PGUSER=enterpriseDB
export PATH

~.
~.
~.
~.
~.
~.

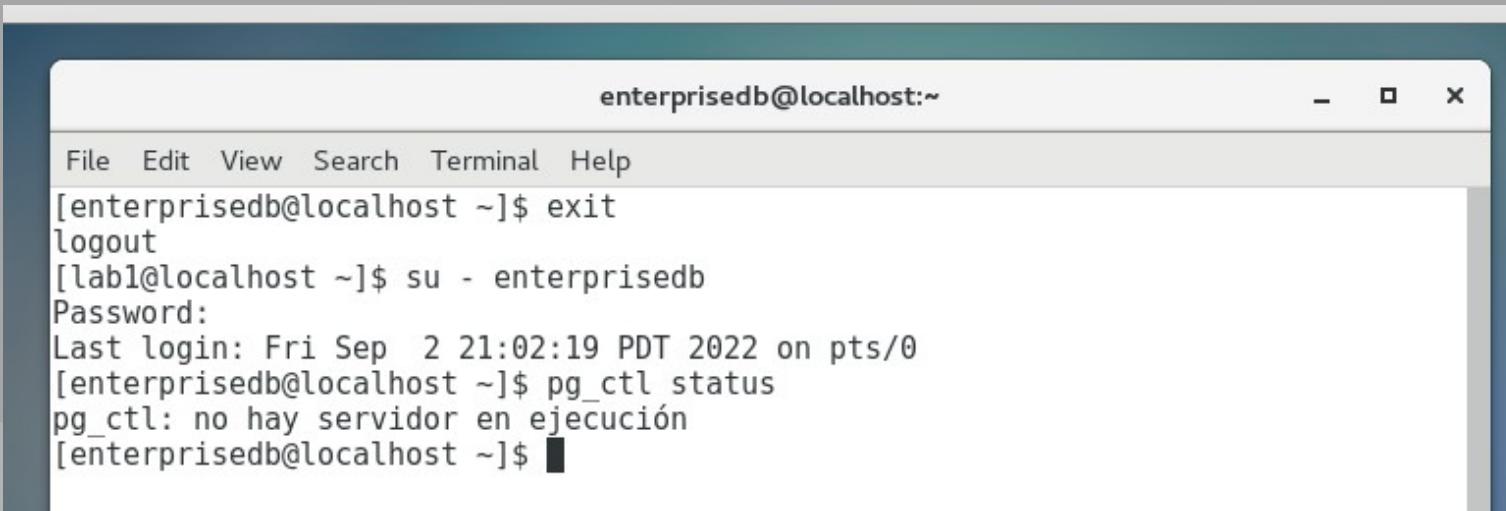
".bash_profile" 15L, 315C
```

LABORATORIO 1.

Instalación

Configuración de Variables del usuario enterpriseDB

- Probamos la nueva configuración:
 - [enterpriseDB@localhost ~]\$ exit
 - logout
 - [lab1@localhost ~]\$ su enterpriseDB
 - Password:
 - Last login: Fri Sep 2 21:02:19 PDT 2022 on pts/0
 - [enterpriseDB@localhost ~]\$ pg_ctl status
 - pg_ctl: no hay servidor en ejecución
 - [enterpriseDB@localhost ~]\$



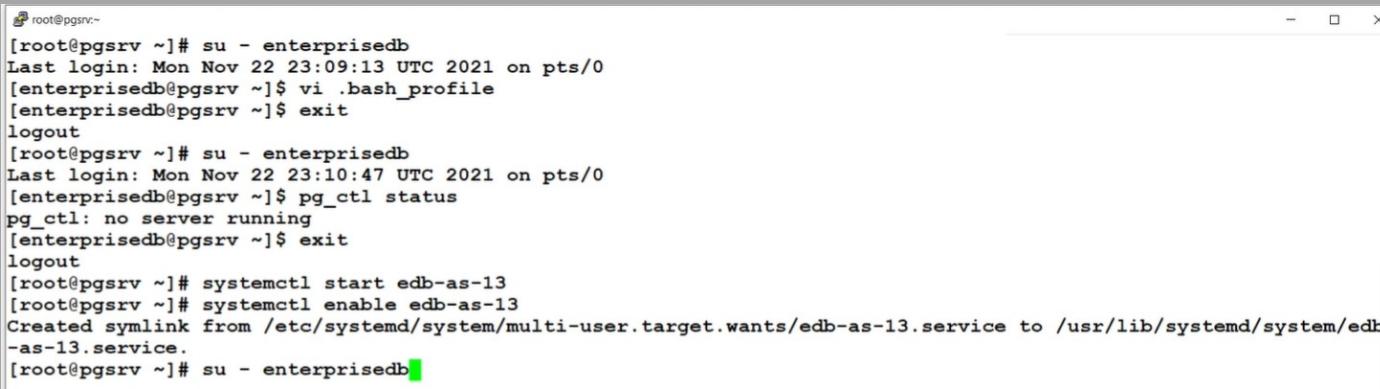
```
enterpriseDB@localhost:~  
File Edit View Search Terminal Help  
[enterpriseDB@localhost ~]$ exit  
logout  
[lab1@localhost ~]$ su - enterpriseDB  
Password:  
Last login: Fri Sep 2 21:02:19 PDT 2022 on pts/0  
[enterpriseDB@localhost ~]$ pg_ctl status  
pg_ctl: no hay servidor en ejecución  
[enterpriseDB@localhost ~]$ █
```

LABORATORIO 1.

Instalación

Configuración de Variables del usuario enterpriseDB

- Observamos que no hay ningún servidor de Postgres en ejecución así que haremos lo siguiente:
 - [enterpriseDB@pgsrv ~]\$ exit
 - Logout (Debemos logearnos como root)
 - [root@pgsrv ~]# systemctl start edb-as-13
 - [root@pgsrv ~]# systemctl enable edb-as-13
 - Created symlink from /etc/systemd/system/multi-user.target.wants/edb-as-13.service to /usr/lib/systemd/system/edb-as-13.service.
 - [root@pgsrv ~]# su - enterpriseDB]



```
root@pgsrv:~# su - enterpriseDB
Last login: Mon Nov 22 23:09:13 UTC 2021 on pts/0
[enterpriseDB@pgsrv ~]$ vi .bash_profile
[enterpriseDB@pgsrv ~]$ exit
logout
[root@pgsrv ~]# su - enterpriseDB
Last login: Mon Nov 22 23:10:47 UTC 2021 on pts/0
[enterpriseDB@pgsrv ~]$ pg_ctl status
pg_ctl: no server running
[enterpriseDB@pgsrv ~]$ exit
logout
[root@pgsrv ~]# systemctl start edb-as-13
[root@pgsrv ~]# systemctl enable edb-as-13
Created symlink from /etc/systemd/system/multi-user.target.wants/edb-as-13.service to /usr/lib/systemd/system/edb-as-13.service.
[root@pgsrv ~]# su - enterpriseDB
```

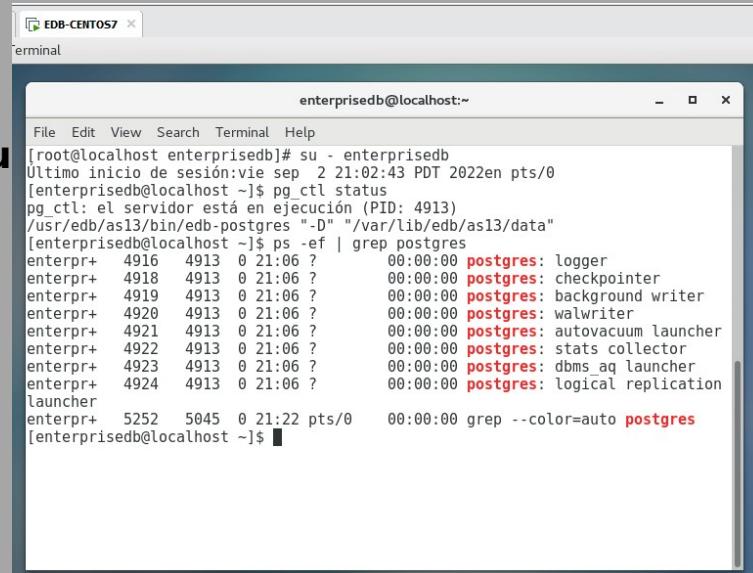
LABORATORIO 1.

Instalación

Configuración de Variables del usuario enterpriseDB.

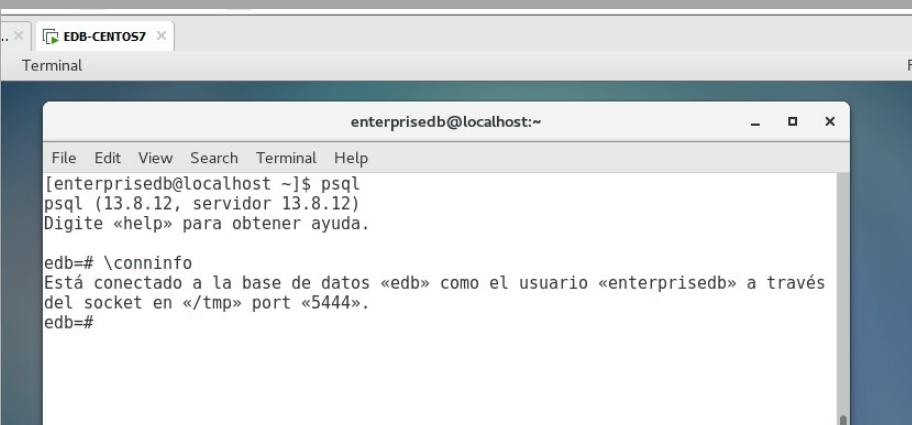
- **NOTA: Puede ser (su -enterprise) o (su -enterprisedb)**

Verificamos de nuevo si el servidor esta corriendo:



```
[root@localhost enterpriseDB]# su - enterpriseDB
Último inicio de sesión: vie sep 2 21:02:43 PDT 2022en pts/0
[enterpriseDB@localhost ~]$ pg_ctl status
pg_ctl: el servidor está en ejecución (PID: 4913)
/usr/edb/as13/bin/edb-postgres "-D" "/var/lib/edb/as13/data"
[enterpriseDB@localhost ~]$ ps -ef | grep postgres
enterpr+ 4916 4913 0 21:06 ? 00:00:00 postgres: logger
enterpr+ 4918 4913 0 21:06 ? 00:00:00 postgres: checkpointer
enterpr+ 4919 4913 0 21:06 ? 00:00:00 postgres: background writer
enterpr+ 4920 4913 0 21:06 ? 00:00:00 postgres: walwriter
enterpr+ 4921 4913 0 21:06 ? 00:00:00 postgres: autovacuum launcher
enterpr+ 4922 4913 0 21:06 ? 00:00:00 postgres: stats collector
enterpr+ 4923 4913 0 21:06 ? 00:00:00 postgres: dbms ad launcher
enterpr+ 4924 4913 0 21:06 ? 00:00:00 postgres: logical replication
launcher
enterpr+ 5252 5045 0 21:22 pts/0 00:00:00 grep --color=auto postgres
[enterpriseDB@localhost ~]$
```

- Escribimos: psql
- \conninfo



```
[enterprisedb@localhost ~]$ psql
psql (13.8.12, servidor 13.8.12)
Digite «help» para obtener ayuda.

edb=# \conninfo
Está conectado a la base de datos «edb» como el usuario «enterpriseDB» a través
del socket en «/tmp» port «5444».
edb#
```

LABORATORIO 2.

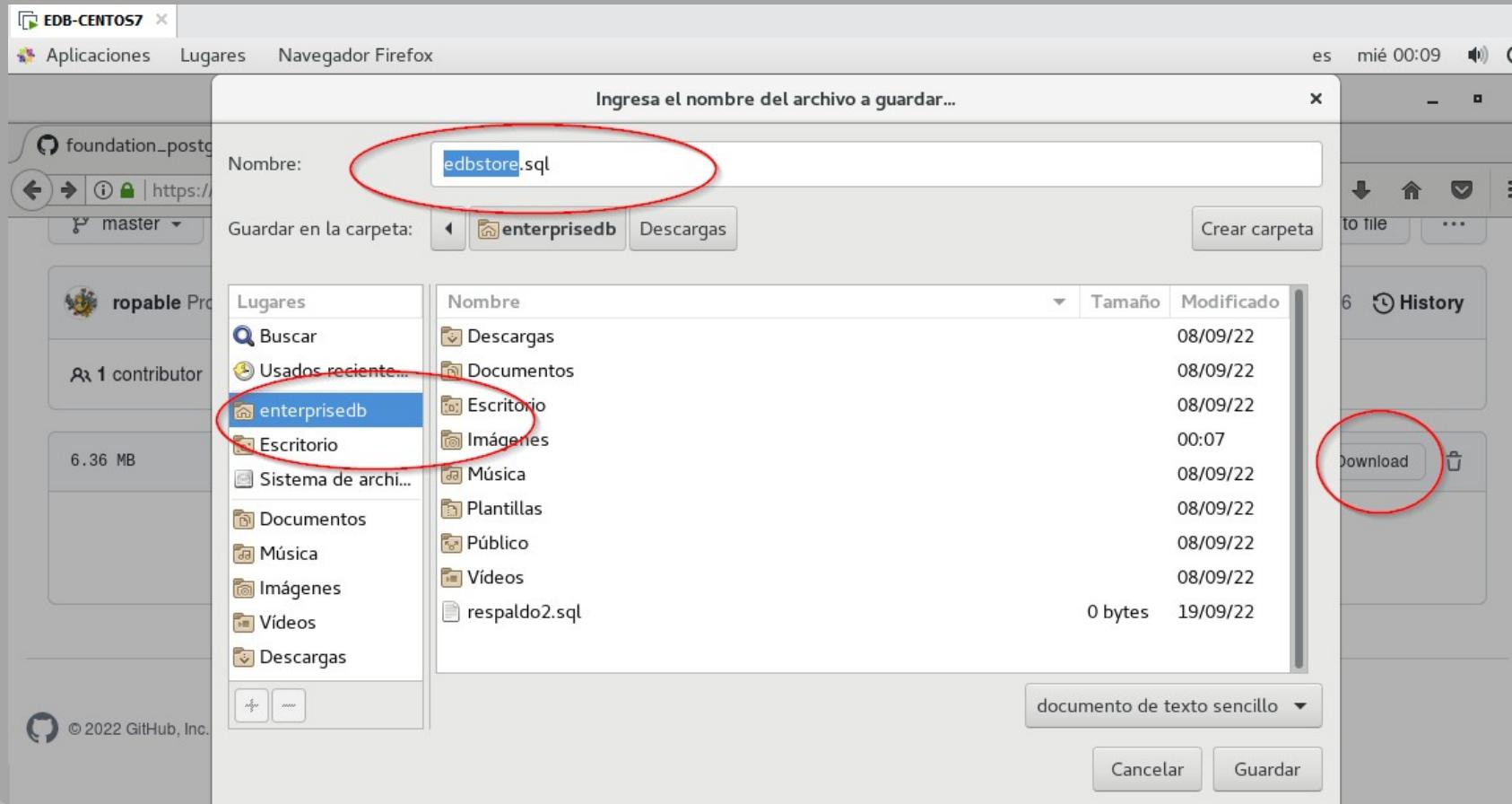
Instalación

- Tu estas trabajando como DBA, un nuevo sitio web necesita ser desarrollado para una tienda de música online.
- Debes de crear el usuario de la base de datos llamado “edbstore” en la base.
- Debes de crear una base de datos llamada “edbstore” con ownership del usuario “edbstore”
- Logueate con el nuevo usuario
- Baja el archivo llamado “edbstore.sql”
- Utiliza psql para ejecutar el script

LABORATORIO 2.

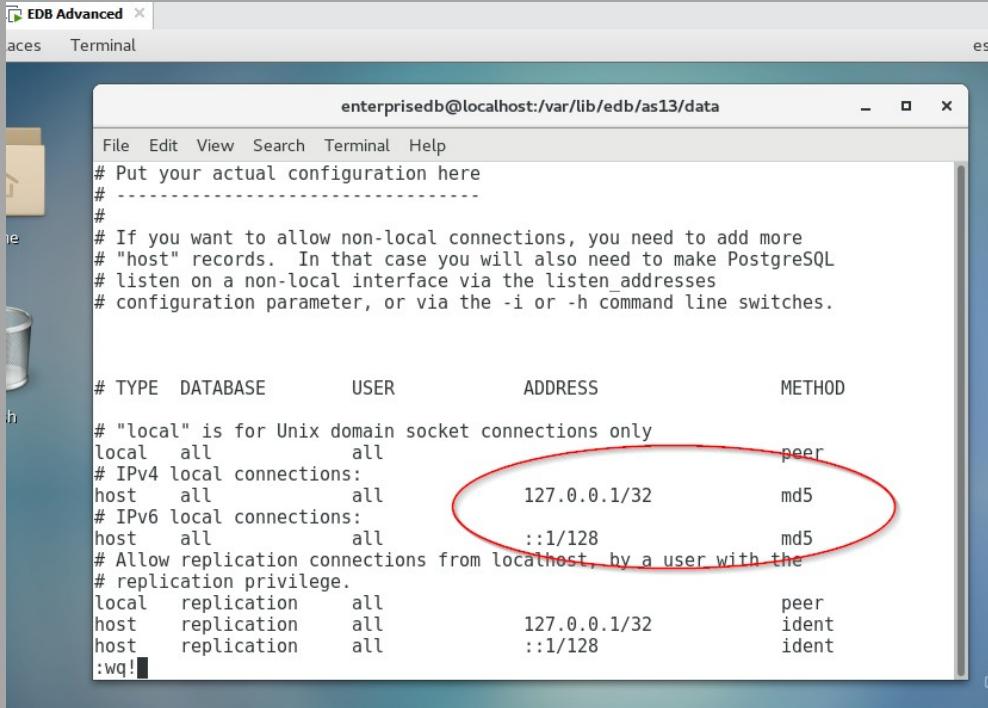
- Vamos a crear la base de datos.
- Baja el archivo “edbstore.sql” a la maquina virtual Postgres.
- https://github.com/ropable/foundation_postgres/blob/master/edbstore.sql
- Botón derecho sobre “Download”.
- Guárdalo en el home del usuario.

LABORATORIO 2.



LABORATORIO 2.

- Cambiamos a MD5 dentro del pg_hba.conf
- Reiniciamos nuestra base de datos.
 - [enterprisedb@localhost data]\$ sudo systemctl stop edb-as-13.service
 - [enterprisedb@localhost data]\$ sudo systemctl start edb-as-13.service

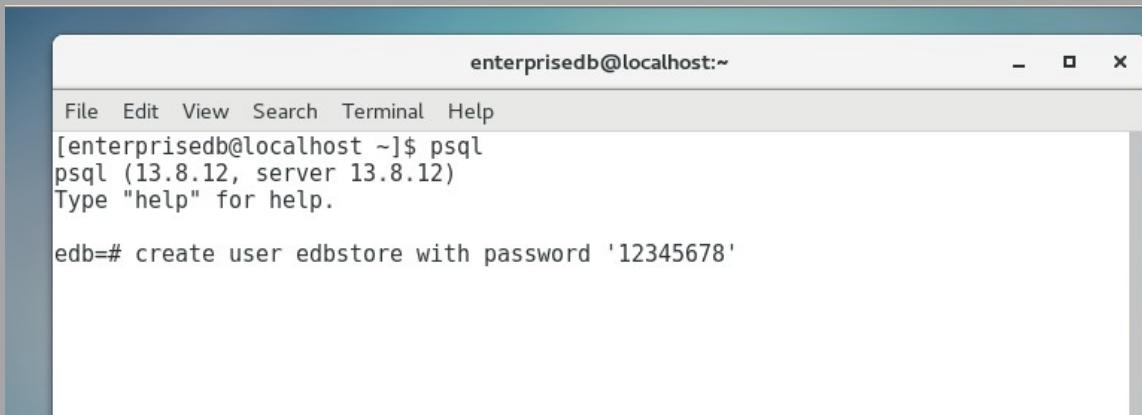


```
edb Advanced x |aces Terminal es
enterprisedb@localhost:/var/lib/edb/as13/data - x
File Edit View Search Terminal Help
# Put your actual configuration here
# -----
#
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.

# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 ident
host replication all ::1/128 ident
:wq!
```

LABORATORIO 2.

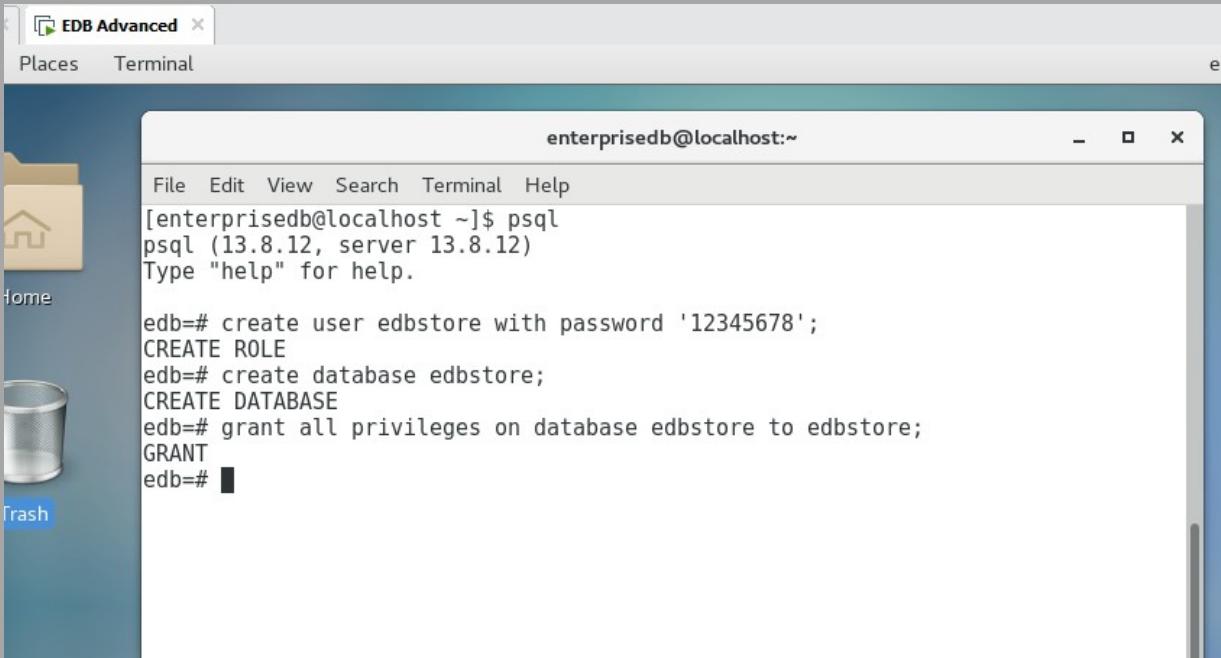
Creamos el usuario:



```
enterprisedb@localhost:~  
File Edit View Search Terminal Help  
[enterprisedb@localhost ~]$ psql  
psql (13.8.12, server 13.8.12)  
Type "help" for help.  
  
edb=# create user edbstore with password '12345678'
```

LABORATORIO 2.

Creamos la base :

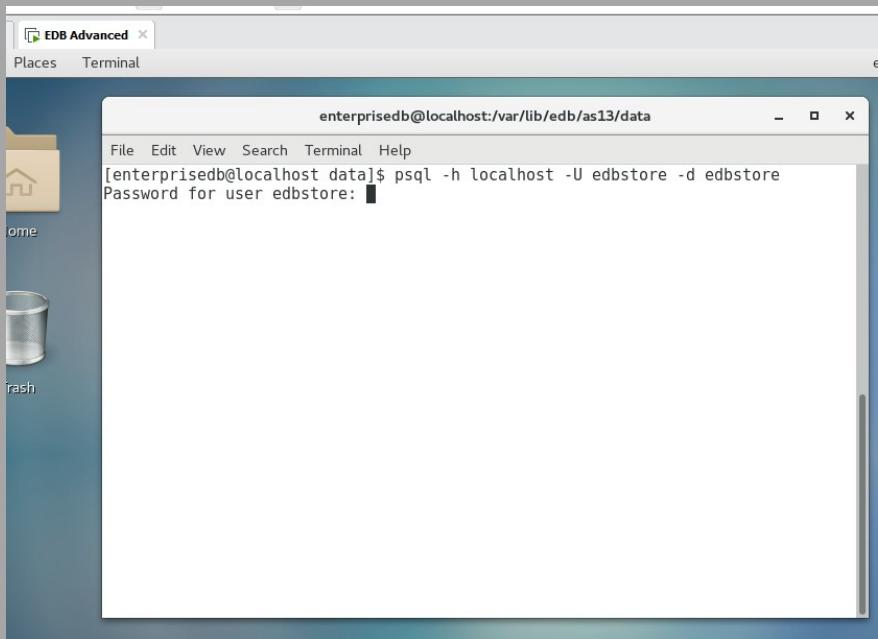


```
enterprisedb@localhost:~$ psql
psql (13.8.12, server 13.8.12)
Type "help" for help.

edb=# create user edbstore with password '12345678';
CREATE ROLE
edb=# create database edbstore;
CREATE DATABASE
edb=# grant all privileges on database edbstore to edbstore;
GRANT
edb=#
```

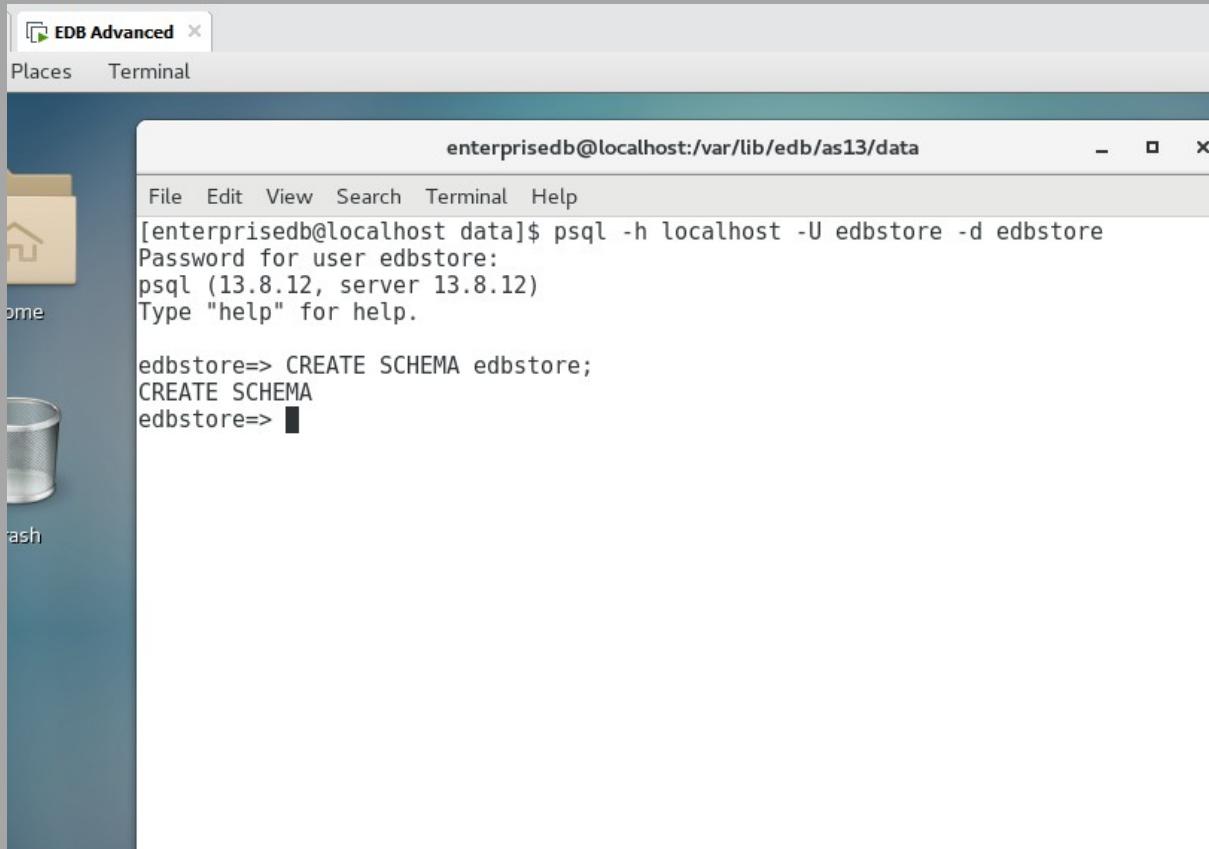
LABORATORIO 2.

- Nos logeamos a la base de datos con el usuario “edbuser”
- Pass: “12345678”



LABORATORIO 2.

Creamos el esquema:



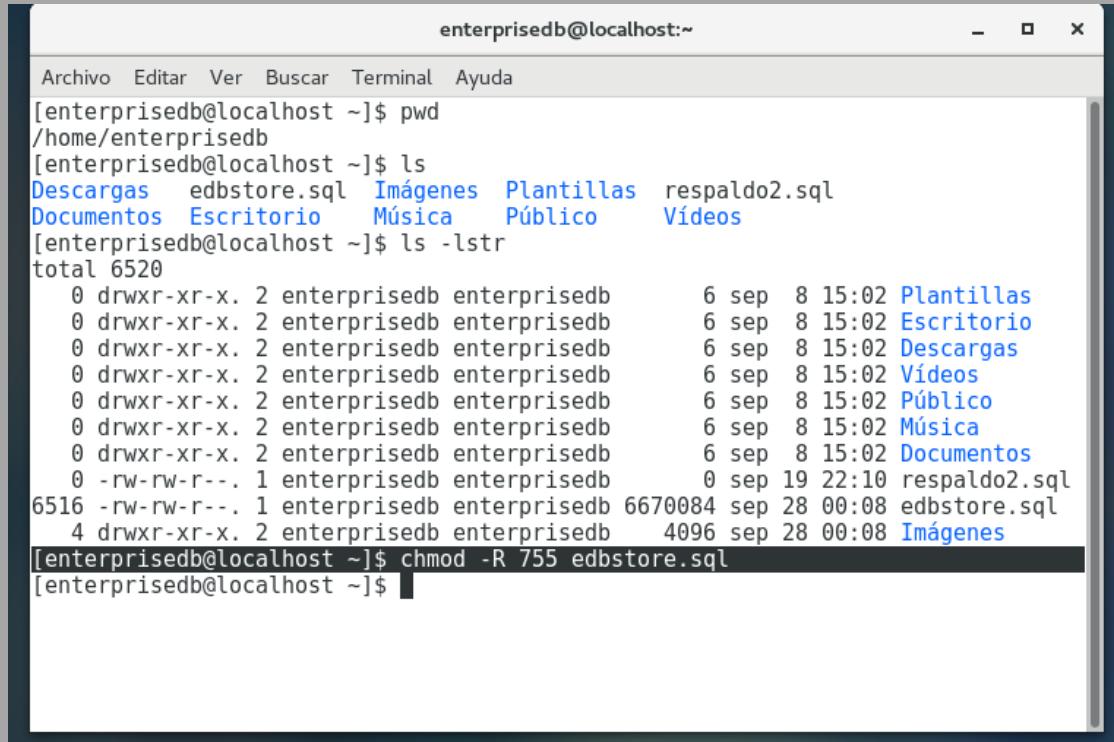
```
enterprisedb@localhost:/var/lib/edb/as13/data
File Edit View Search Terminal Help
[enterprisedb@localhost data]$ psql -h localhost -U edbstore -d edbstore
Password for user edbstore:
psql (13.8.12, server 13.8.12)
Type "help" for help.

edbstore=> CREATE SCHEMA edbstore;
CREATE SCHEMA
edbstore=>
```

LABORATORIO 2.

Dale permisos al archivo.

- [enterprisedb@localhost ~]\$ chmod -R 755 edbstore.sql



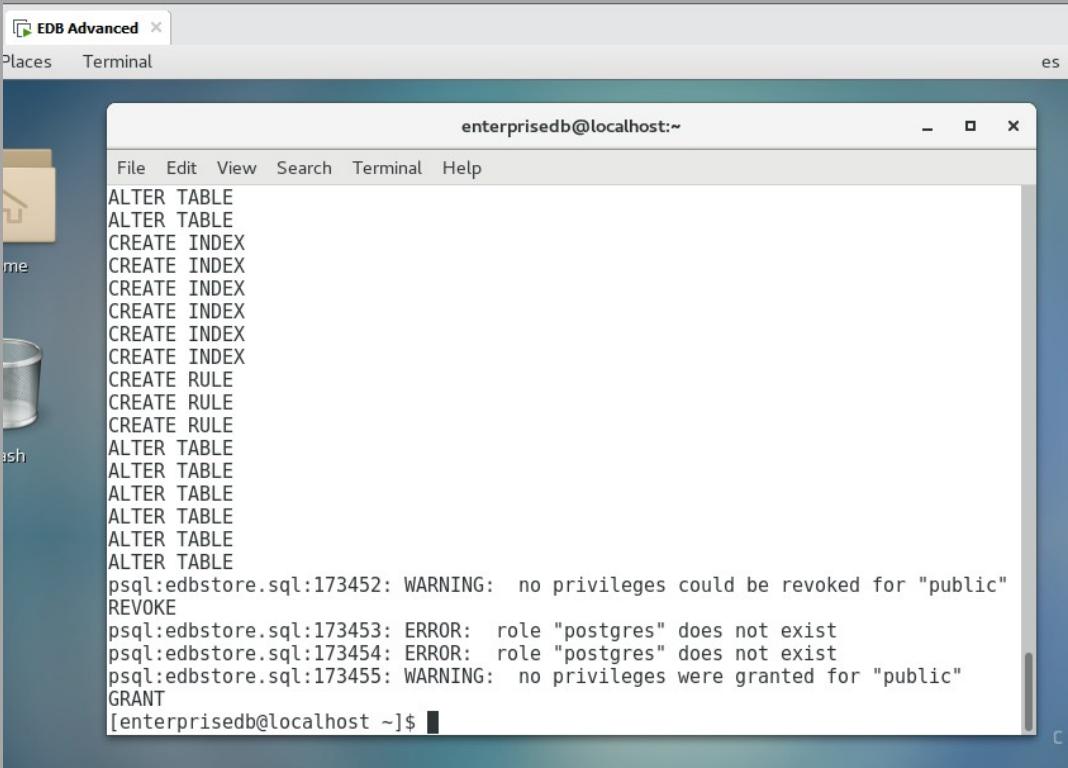
```
enterprisedb@localhost:~
```

```
Archivo Editar Ver Buscar Terminal Ayuda
[enterprisedb@localhost ~]$ pwd
/home/enterprisedb
[enterprisedb@localhost ~]$ ls
Descargas   edbstore.sql  Imágenes  Plantillas  respaldo2.sql
Documentos  Escritorio   Música    Público    Vídeos
[enterprisedb@localhost ~]$ ls -lstr
total 6520
  0 drwxr-xr-x. 2 enterprisedb enterprisedb      6 sep  8 15:02 Plantillas
  0 drwxr-xr-x. 2 enterprisedb enterprisedb      6 sep  8 15:02 Escritorio
  0 drwxr-xr-x. 2 enterprisedb enterprisedb      6 sep  8 15:02 Descargas
  0 drwxr-xr-x. 2 enterprisedb enterprisedb      6 sep  8 15:02 Vídeos
  0 drwxr-xr-x. 2 enterprisedb enterprisedb      6 sep  8 15:02 Público
  0 drwxr-xr-x. 2 enterprisedb enterprisedb      6 sep  8 15:02 Música
  0 drwxr-xr-x. 2 enterprisedb enterprisedb      6 sep  8 15:02 Documentos
  0 -rw-rw-r--. 1 enterprisedb enterprisedb      0 sep 19 22:10 respaldo2.sql
6516 -rw-rw-r--. 1 enterprisedb enterprisedb 6670084 sep 28 00:08 edbstore.sql
  4 drwxr-xr-x. 2 enterprisedb enterprisedb     4096 sep 28 00:08 Imágenes
[enterprisedb@localhost ~]$ chmod -R 755 edbstore.sql
[enterprisedb@localhost ~]$
```

LABORATORIO 2.

Ejecutamos:

- [enterprisedb@localhost ~]\$ psql -h localhost -U edbstore -d edbstore -f edbstore.sql

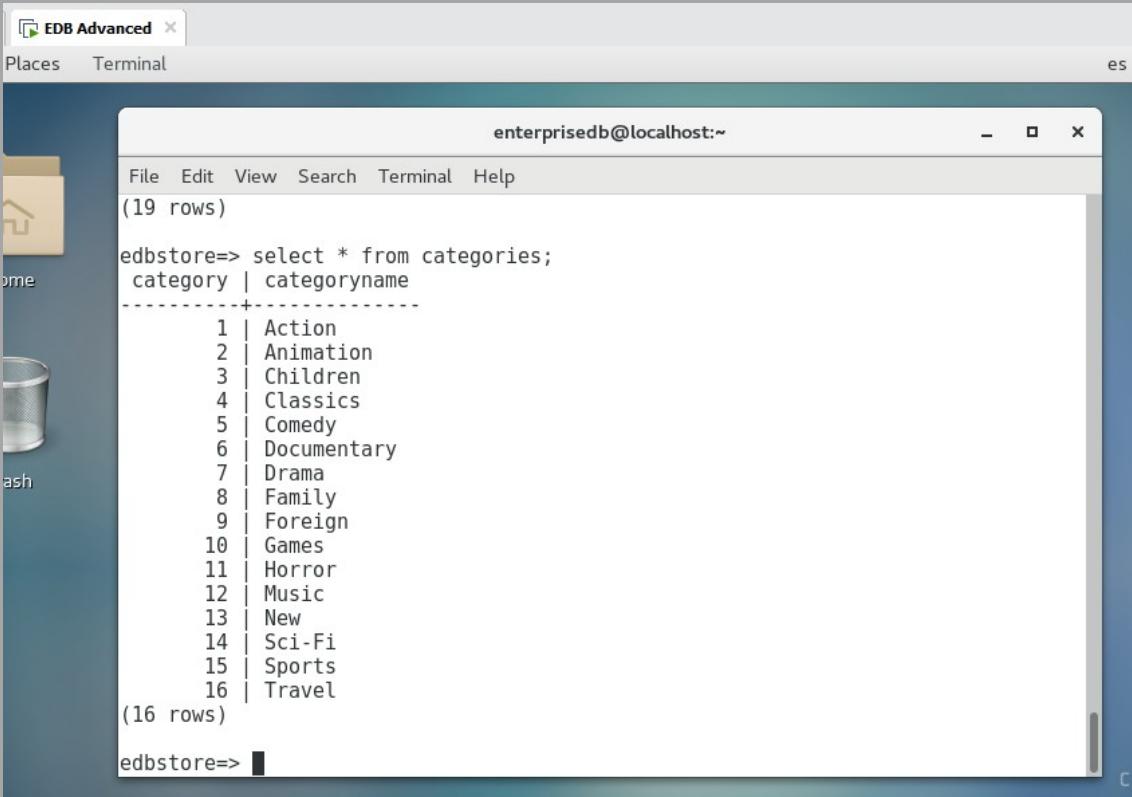


```
edb Advanced x
Places Terminal
enterprisedb@localhost:~ es
File Edit View Search Terminal Help
ALTER TABLE
ALTER TABLE
CREATE INDEX
CREATE RULE
CREATE RULE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
psql:edbstore.sql:173452: WARNING:  no privileges could be revoked for "public"
REVOKE
psql:edbstore.sql:173453: ERROR:  role "postgres" does not exist
psql:edbstore.sql:173454: ERROR:  role "postgres" does not exist
psql:edbstore.sql:173455: WARNING:  no privileges were granted for "public"
GRANT
[enterprisedb@localhost ~]$
```

LABORATORIO 2.

Verificamos que ya tengamos nuestra base de datos:

- [enterprisedb@localhost ~]\$ psql -h localhost -U edbstore -d edbstore



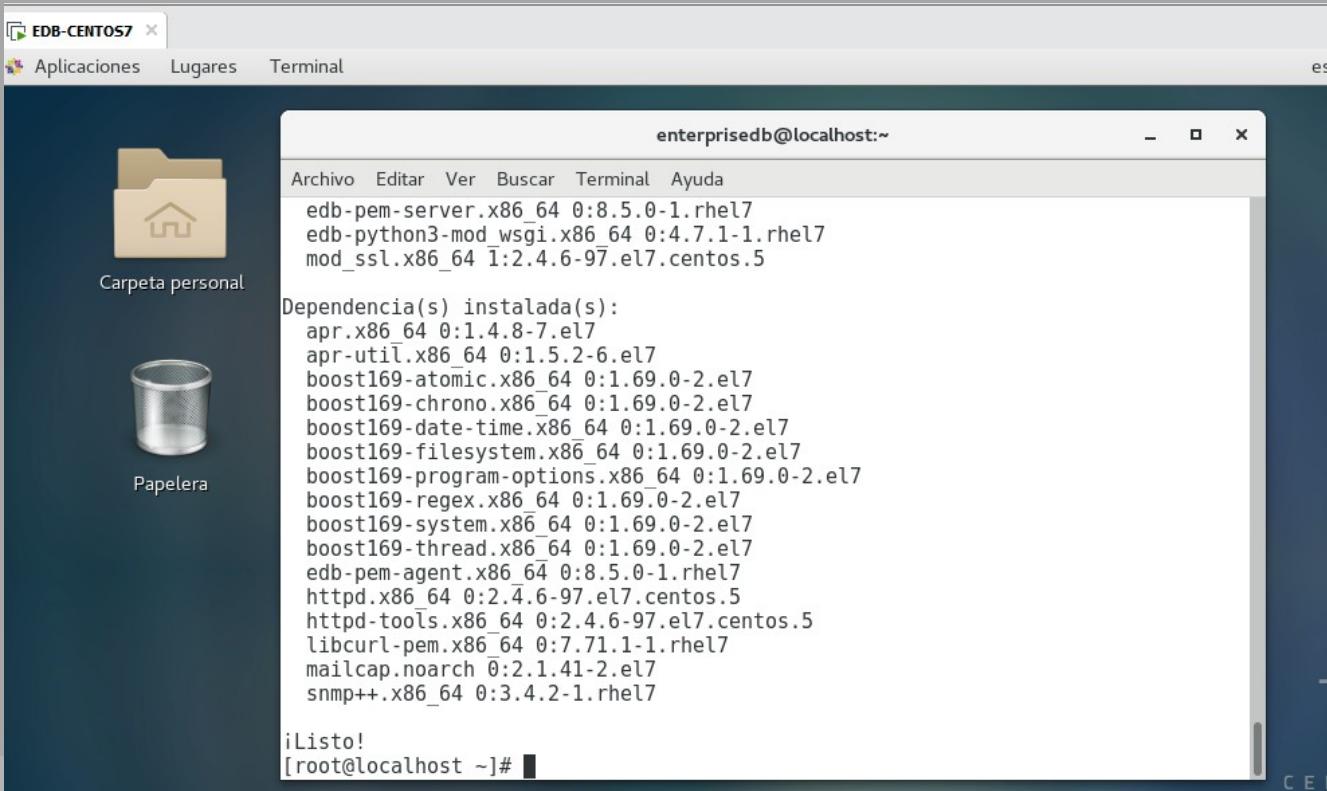
```
edbstore@localhost:~$ psql -h localhost -U edbstore -d edbstore
edbstore=>
edbstore=> select * from categories;
category | categoryname
-----+-----
 1 | Action
 2 | Animation
 3 | Children
 4 | Classics
 5 | Comedy
 6 | Documentary
 7 | Drama
 8 | Family
 9 | Foreign
10 | Games
11 | Horror
12 | Music
13 | New
14 | Sci-Fi
15 | Sports
16 | Travel
(16 rows)
edbstore=>
```

LABORATORIO 3.

Instalación

INSTALAR SERVER PEM

- sudo yum install wxBase mod_wsgi mod_ssl edb-pem-server -y



```
edb@localhost:~$ sudo yum install wxBase mod_wsgi mod_ssl edb-pem-server -y
edb-pem-server.x86_64 0:8.5.0-1.rhel7
edb-python3-mod_wsgi.x86_64 0:4.7.1-1.rhel7
mod_ssl.x86_64 1:2.4.6-97.el7.centos.5

Dependencia(s) instalada(s):
apr.x86_64 0:1.4.8-7.el7
apr-util.x86_64 0:1.5.2-6.el7
boost169-atomic.x86_64 0:1.69.0-2.el7
boost169-chrono.x86_64 0:1.69.0-2.el7
boost169-date-time.x86_64 0:1.69.0-2.el7
boost169-filesystem.x86_64 0:1.69.0-2.el7
boost169-program-options.x86_64 0:1.69.0-2.el7
boost169-regex.x86_64 0:1.69.0-2.el7
boost169-system.x86_64 0:1.69.0-2.el7
boost169-thread.x86_64 0:1.69.0-2.el7
edb-pem-agent.x86_64 0:8.5.0-1.rhel7
httpd.x86_64 0:2.4.6-97.el7.centos.5
httpd-tools.x86_64 0:2.4.6-97.el7.centos.5
libcurl-pem.x86_64 0:7.71.1-1.rhel7
mailcap.noarch 0:2.1.41-2.el7
snmp++.x86_64 0:3.4.2-1.rhel7

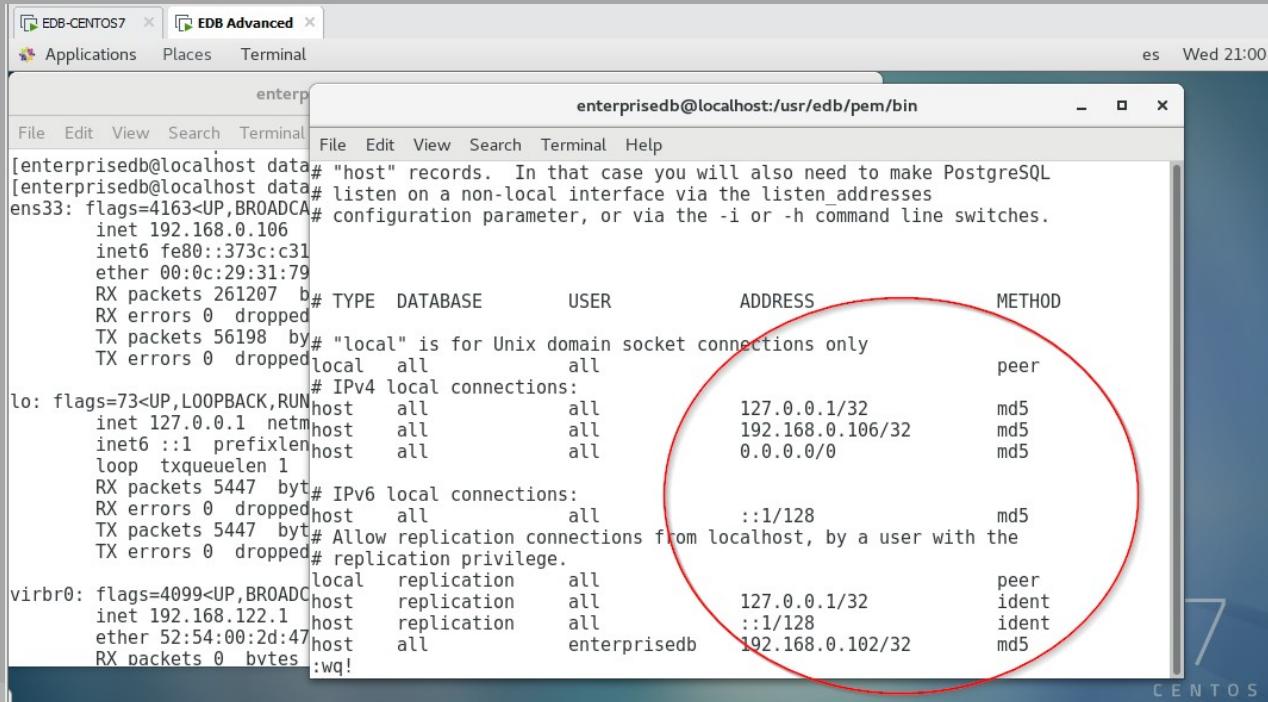
[  Listo!
[root@localhost ~]#
```

LABORATORIO 2.

- Cambiamos a MD5 dentro del pg_hba.conf.
sudo vi /var/lib/edb/as13/data.

- Reiniciamos nuestra base de datos.

```
[enterprisedb@localhost data]$ sudo systemctl stop edb-as-13.service  
[enterprisedb@localhost data]$ sudo systemctl start edb-as-13.service
```



The screenshot shows a terminal window titled "edb@localhost:/usr/edb/pem/bin" displaying the contents of the pg_hba.conf file. A red circle highlights the line where md5 authentication is configured for a host connection:

TYPE	DATABASE	USER	ADDRESS	METHOD
host	all	all	127.0.0.1/32	md5
host	all	all	192.168.0.106/32	md5
host	all	all	0.0.0.0/0	md5

The rest of the file contains configuration for local connections and replication connections.

LABORATORIO 3.

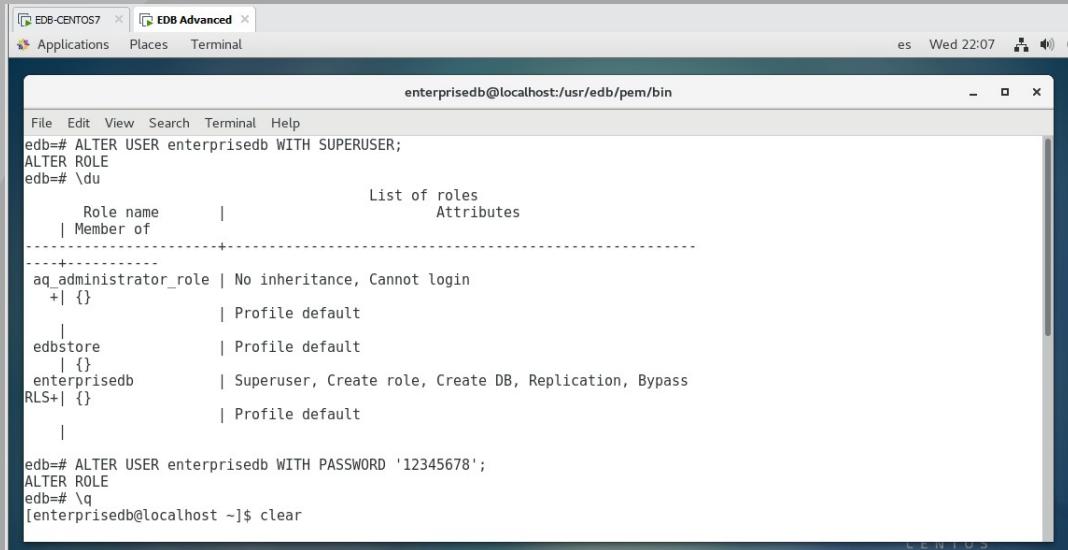
Instalación

CONFIGURACION PEM

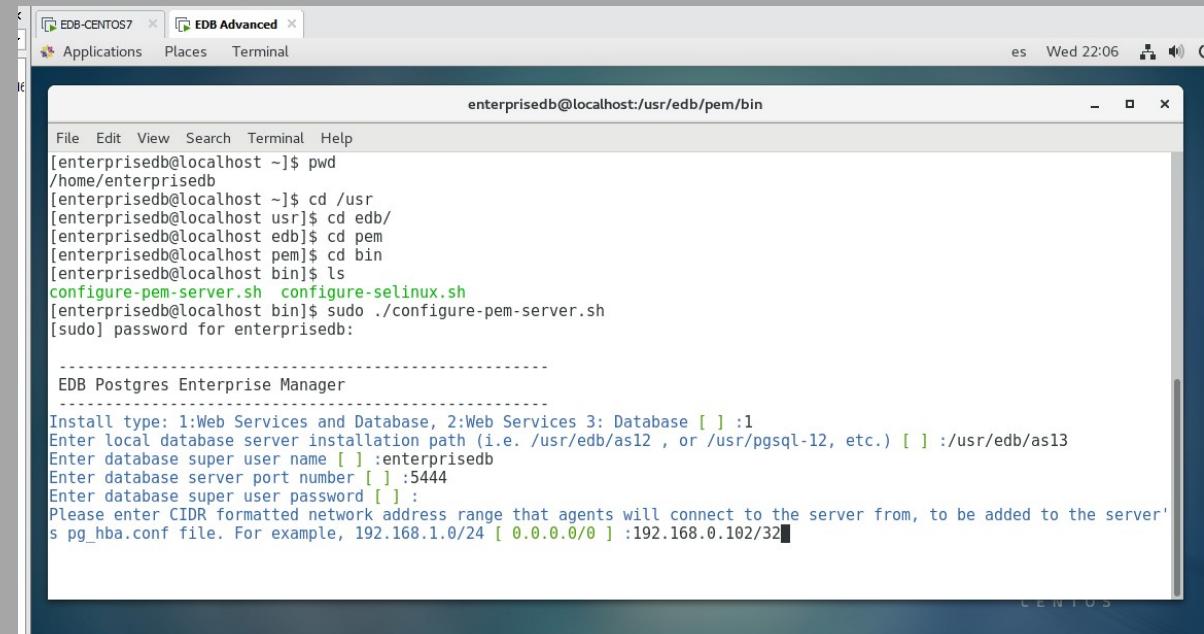
- Entramos a
 - /usr/edb/pem/bin:
- Ejecutamos el script de configuración : ./configure-pem-server.sh
- Tecleamos 1
- /usr/edb/as13/
- enterprisedb
- 5444
- 12345678
- 192.168.0.102/32 (IP MAQUINA)
- edb-as-13

LABORATORIO 3.

Instalación



```
edb=# ALTER USER enterpriseedb WITH SUPERUSER;
ALTER ROLE
edb=# \du
      List of roles
Role name | Attributes
| Member of
-----+-----
ag_administrator_role | No inheritance, Cannot login
+| {}
| Profile default
edbstore | Profile default
+| {}
enterpriseedb | Superuser, Create role, Create DB, Replication, Bypass
RLS+| {}
| Profile default
|
edb=# ALTER USER enterpriseedb WITH PASSWORD '12345678';
ALTER ROLE
edb=# \q
[enterpriseedb@localhost ~]$ clear
```



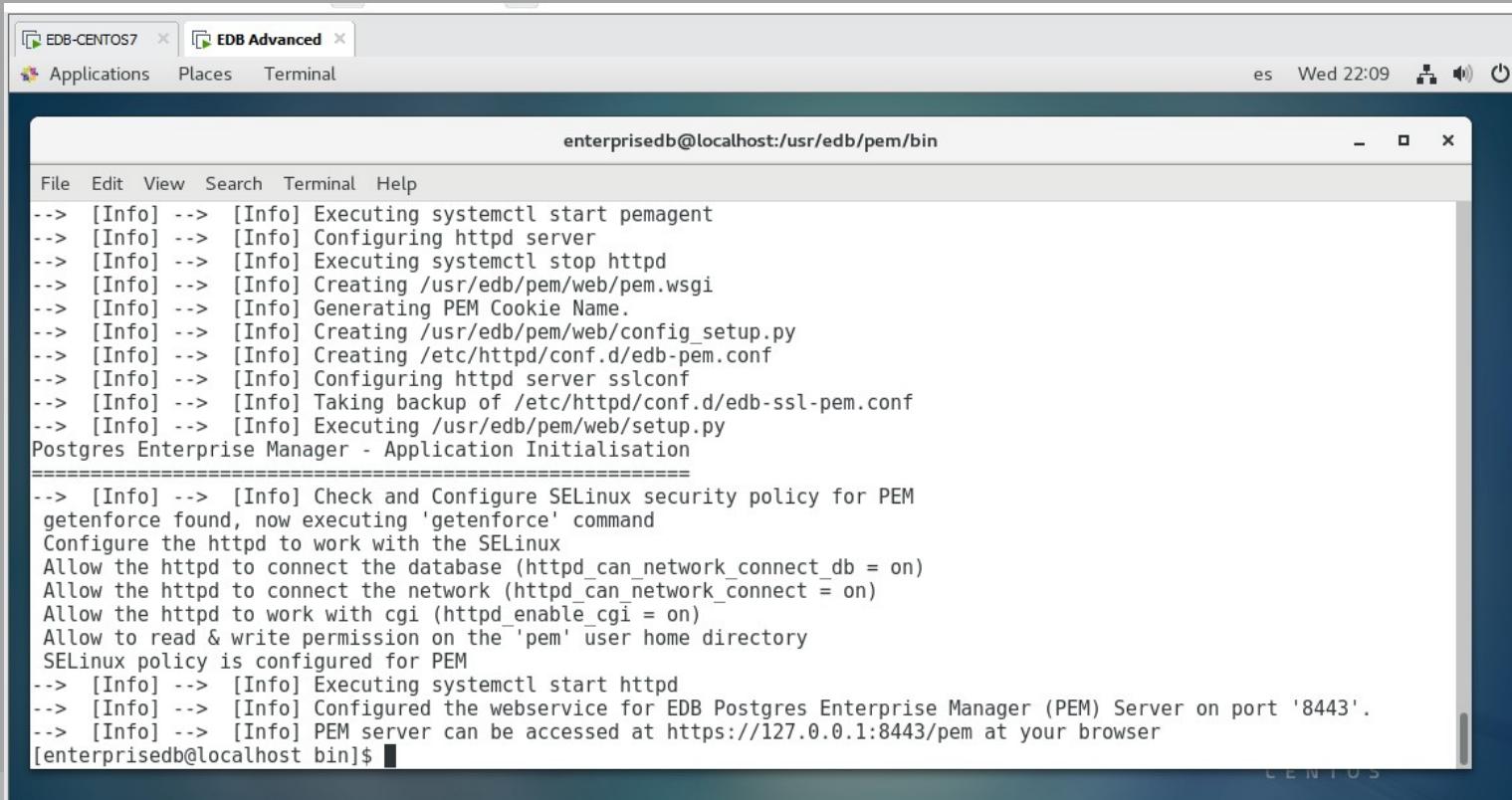
```
[enterpriseedb@localhost ~]$ pwd
/home/enterpriseedb
[enterpriseedb@localhost ~]$ cd /usr
[enterpriseedb@localhost usr]$ cd edb/
[enterpriseedb@localhost edb]$ cd pem
[enterpriseedb@localhost pem]$ cd bin
[enterpriseedb@localhost bin]$ ls
configure-pem-server.sh  configure-selinux.sh
[enterpriseedb@localhost bin]$ sudo ./configure-pem-server.sh
[sudo] password for enterpriseedb:

-----
EDB Postgres Enterprise Manager
-----
Install type: 1:Web Services and Database, 2:Web Services 3: Database [ ] :1
Enter local database server installation path (i.e. /usr/edb/as12 , or /usr/pgsql-12, etc.) [ ] :/usr/edb/as13
Enter database super user name [ ] :enterpriseedb
Enter database server port number [ ] :5444
Enter database super user password [ ] :
Please enter CIDR formatted network address range that agents will connect to the server from, to be added to the server's pg_hba.conf file. For example, 192.168.1.0/24 [ 0.0.0.0/0 ] :192.168.0.102/32
```

LABORATORIO 3.

Instalación

- Finalmente comprobamos la instalación exitosa.



The screenshot shows a terminal window titled "enterprisedb@localhost:/usr/edb/pem/bin" running on a CentOS 7 desktop environment. The window displays the output of a command that performs various configuration steps for the PEM server. Key log entries include:

- Executing systemctl start pemagent
- Configuring httpd server
- Executing systemctl stop httpd
- Creating /usr/edb/pem/web/pem.wsgi
- Generating PEM Cookie Name
- Creating /usr/edb/pem/web/config_setup.py
- Creating /etc/httpd/conf.d/edb-pem.conf
- Configuring httpd server sslconf
- Taking backup of /etc/httpd/conf.d/edb-ssl-pem.conf
- Executing /usr/edb/pem/web/setup.py
- Postgres Enterprise Manager - Application Initialisation
- Checking and configuring SELinux security policy for PEM
- Allowing httpd to connect the database (httpd_can_network_connect_db = on)
- Allowing httpd to connect the network (httpd_can_network_connect = on)
- Allowing httpd to work with cgi (httpd_enable_cgi = on)
- Allowing read & write permission on the 'pem' user home directory
- SELinux policy is configured for PEM
- Executing systemctl start httpd
- Configuring the webservice for EDB Postgres Enterprise Manager (PEM) Server on port '8443'
- PEM server can be accessed at https://127.0.0.1:8443/pem at your browser

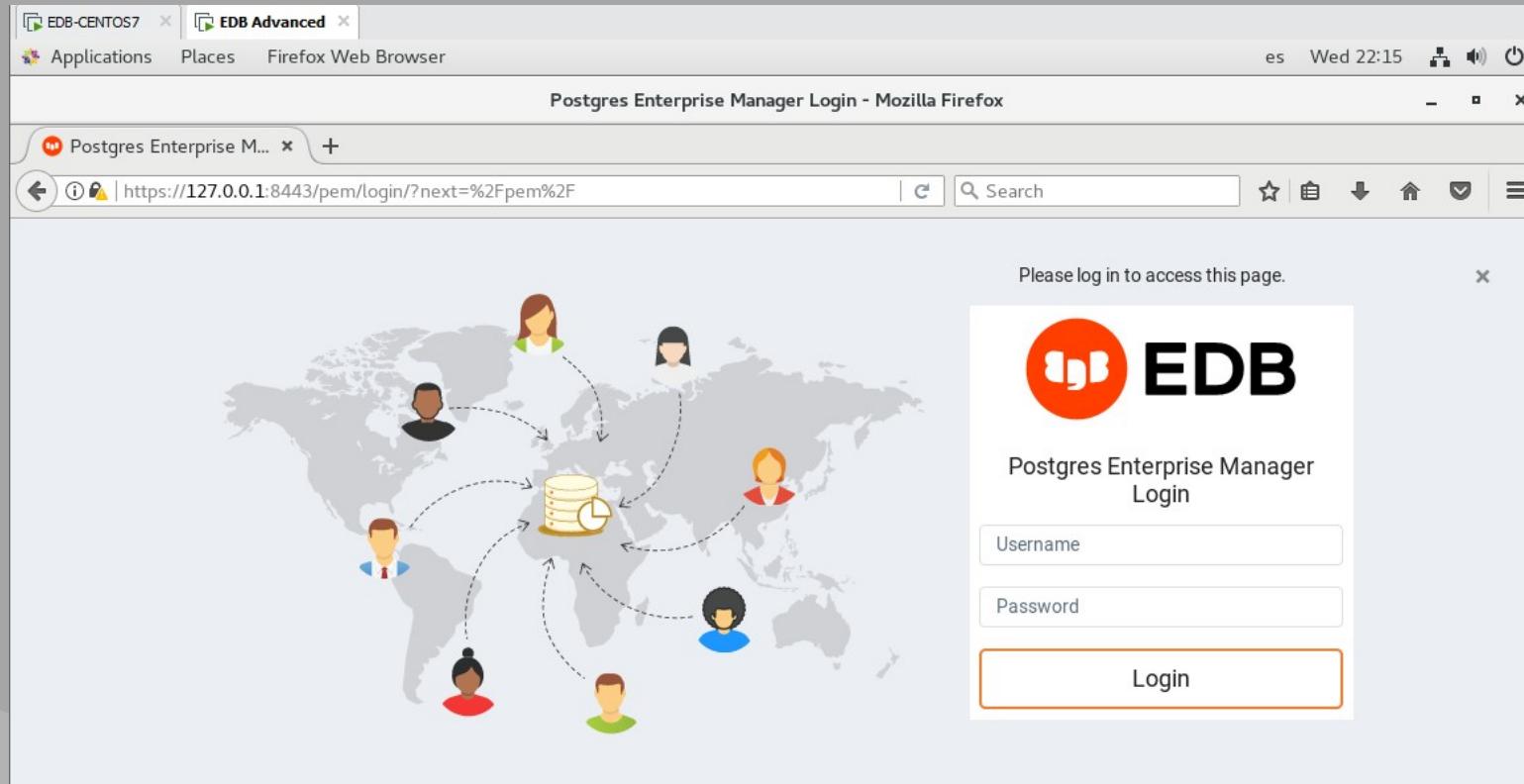
The terminal prompt at the bottom is [enterprisedb@localhost bin]\$.

LABORATORIO 3.

Instalación

Finalmente comprobamos la instalación exitosa.

- **NOTA: No olvides bajar el firewall.**



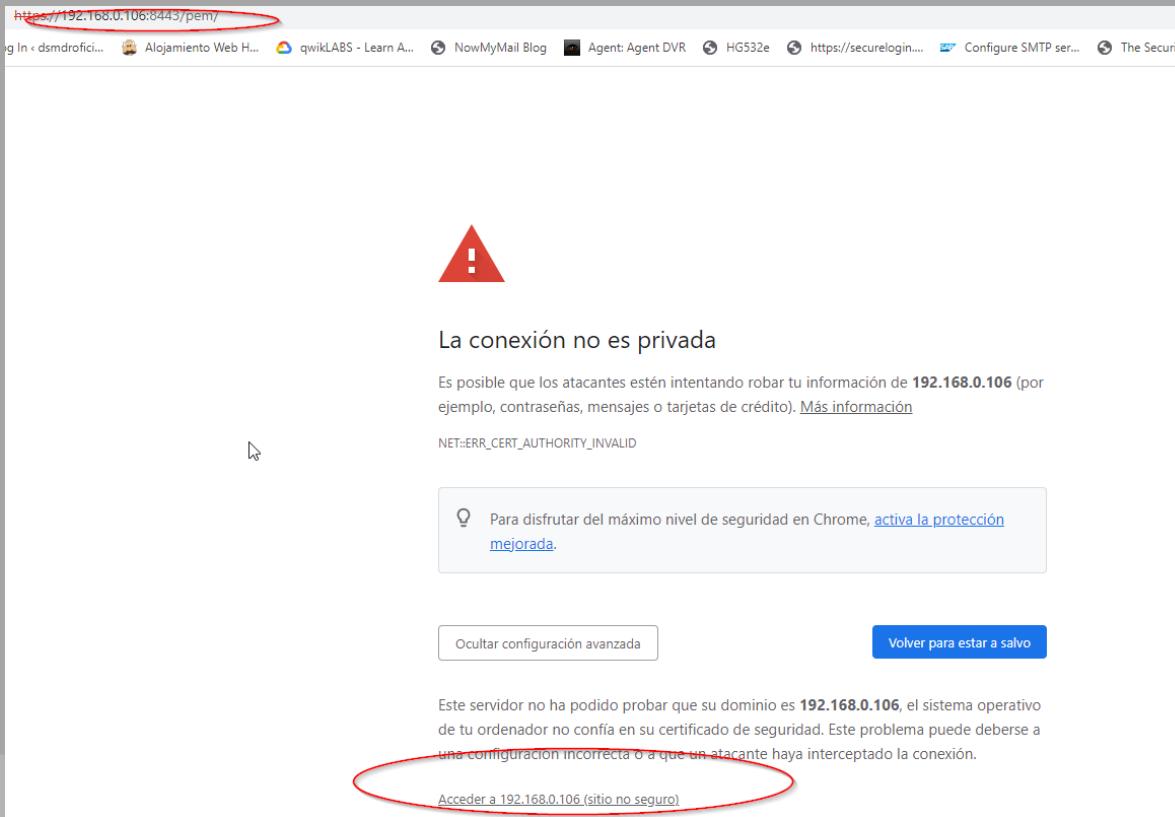
LABORATORIO 3.

Instalación

Finalmente comprobamos la instalación exitosa.

- **NOTA: No olvides bajar el firewall.**

Accesando desde Windows al PEM.



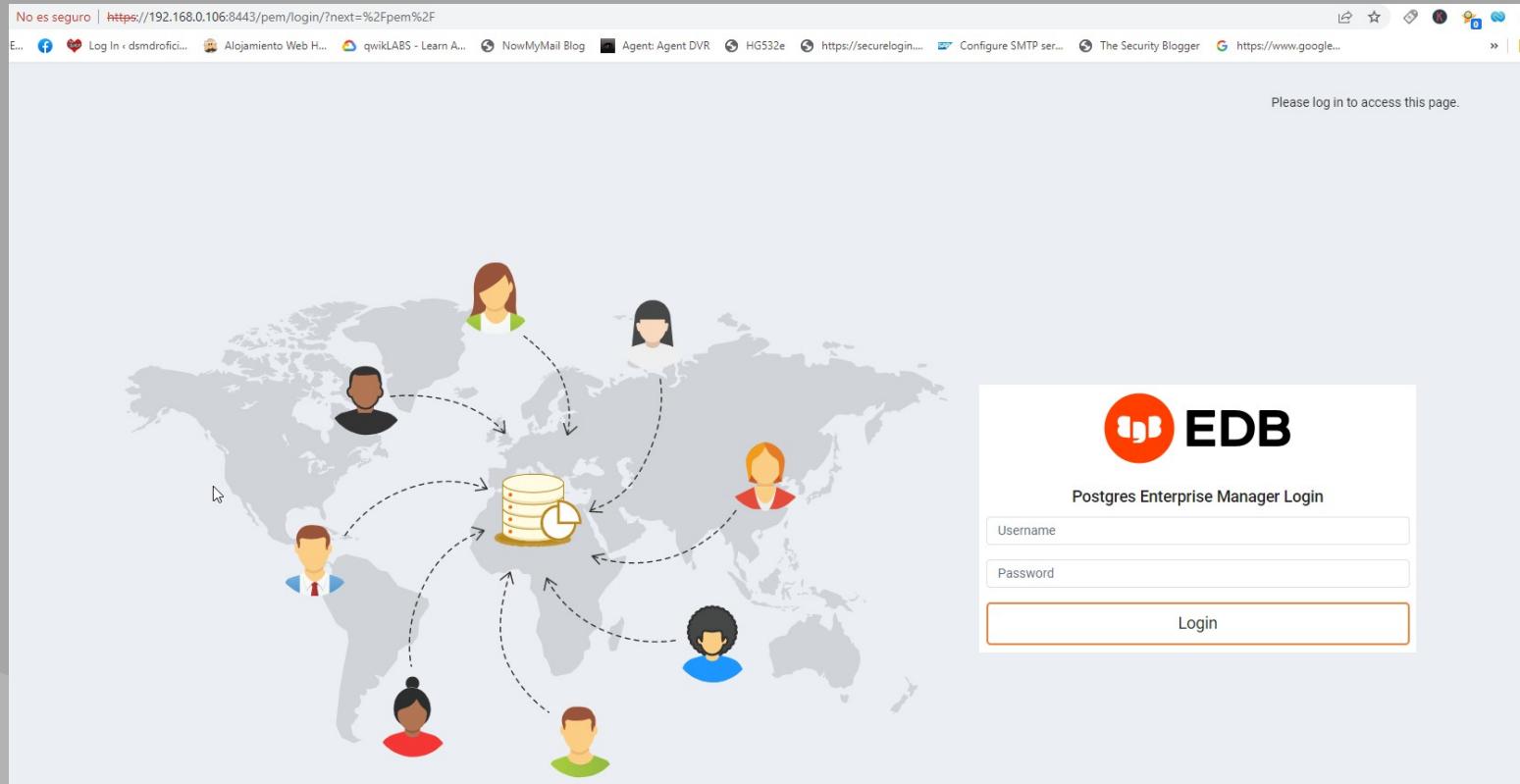
LABORATORIO 3.

Instalación

Finalmente comprobamos la instalación exitosa.

- **NOTA: No olvides bajar el firewall.**

Accesando desde Windows al PEM.



LABORATORIO 3.

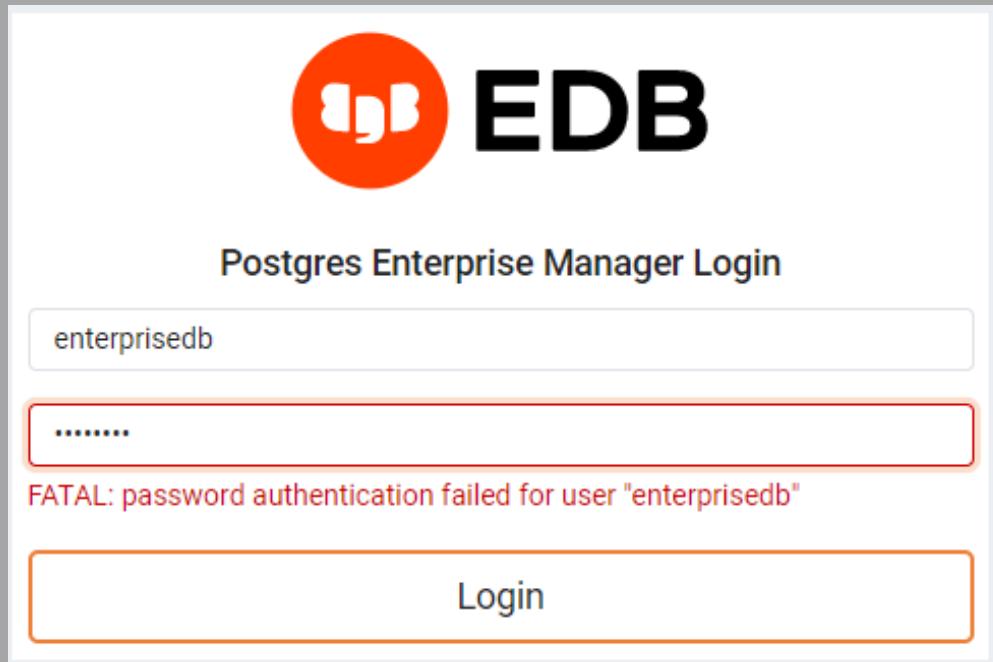
Instalación

Finalmente comprobamos la instalación exitosa.

- **NOTA: No olvides bajar el firewall.**

Accesando desde Windows al PEM.

- Pass: 12345678



LABORATORIO 3.

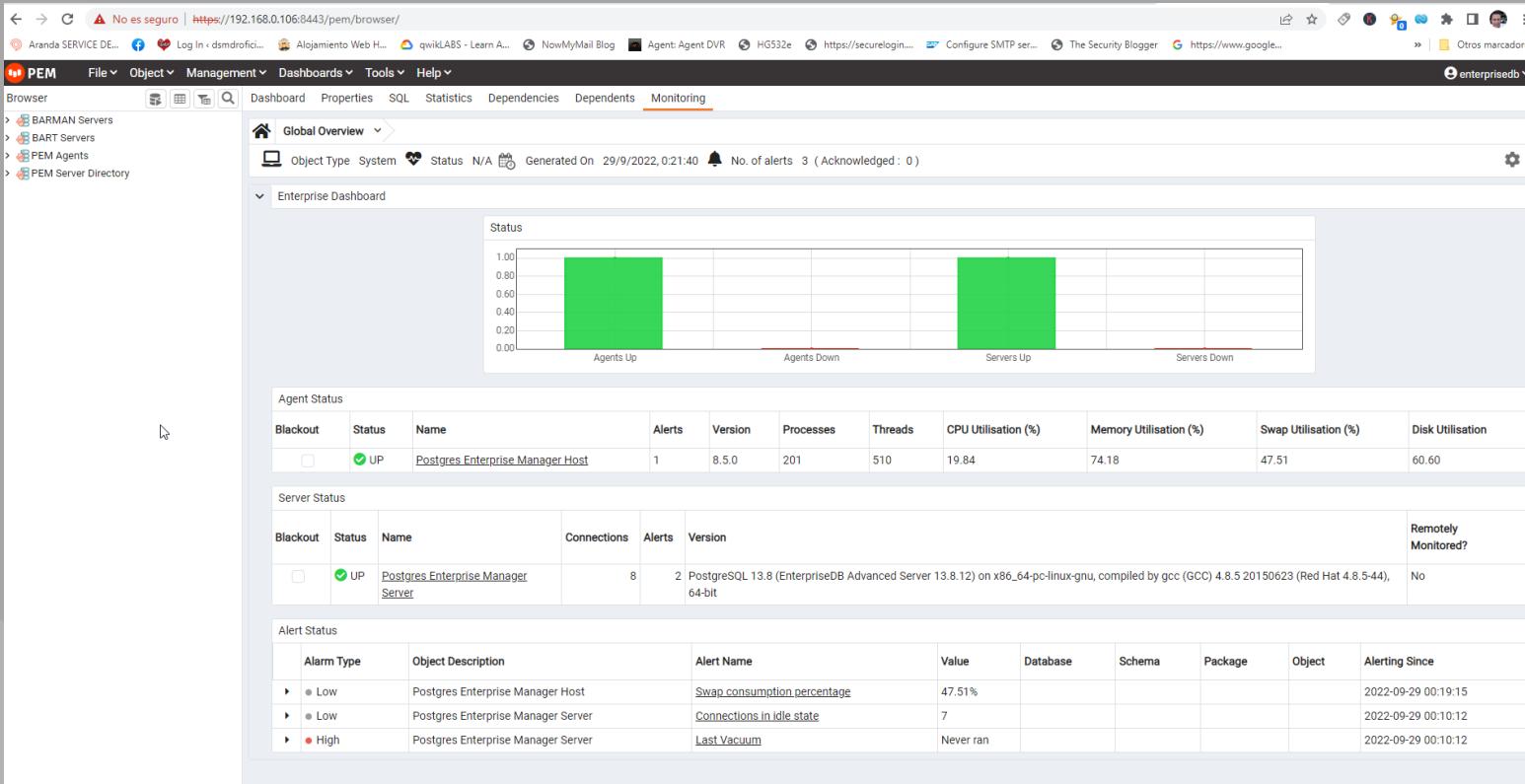
Instalación

Finalmente comprobamos la instalación exitosa.

- NOTA: No olvides bajar el firewall.

Accesando desde Windows al PEM.

- Pass: 12345678
- NOTA: Si haces reboot, no olvides levantar el servicio “httpd” antes de acceder a PEM.



The screenshot shows the PEM web interface with the following sections:

- Global Overview:** Displays a chart titled "Status" comparing "Agents Up" and "Agents Down" (green bars) against "Servers Up" and "Servers Down" (green bars). The chart shows 1.00 for Agents Up, 0.00 for Agents Down, 1.00 for Servers Up, and 0.00 for Servers Down.
- Enterprise Dashboard:** Contains a "Status" chart and a table for "Agent Status".

Blackout	Status	Name	Alerts	Version	Processes	Threads	CPU Utilisation (%)	Memory Utilisation (%)	Swap Utilisation (%)	Disk Utilisation
<input type="checkbox"/>	UP	Postgres Enterprise Manager Host	1	8.5.0	201	510	19.84	74.18	47.51	60.60
- Server Status:** Shows a table for "Server Status".

Blackout	Status	Name	Connections	Alerts	Version	Remotely Monitored?
<input type="checkbox"/>	UP	Postgres Enterprise Manager Server	8	2	PostgreSQL 13.8 (EnterpriseDB Advanced Server 13.8.12) on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44), 64-bit	No
- Alert Status:** Displays a table for "Alert Status".

Alarm Type	Object Description	Alert Name	Value	Database	Schema	Package	Object	Alerting Since
Low	Postgres Enterprise Manager Host	Swap consumption percentage	47.51%					2022-09-29 00:19:15
Low	Postgres Enterprise Manager Server	Connections in Idle state	7					2022-09-29 00:10:12
High	Postgres Enterprise Manager Server	Last Vacuum	Never ran					2022-09-29 00:10:12

PEM QUERYS.

QUERYS DE ADMINISTRACIÓN

- Conexiones Ociosas.
 - Conexiones no ociosas.
 - Conteo de tg_transactions.
 - Eliminación de conexiones con más de un minuto ociosas.
 - Bloqueos actuales tipos.
 - Bloqueos actuales tablas bloqueadas.
 - Tamaño tablas BBDD.
 - Crear tablas sin datos y truncar.
 - Creación de backups.
-
- **NOTA: Ver archivo SQL Administracion.sql**

LABORATORIO 4.

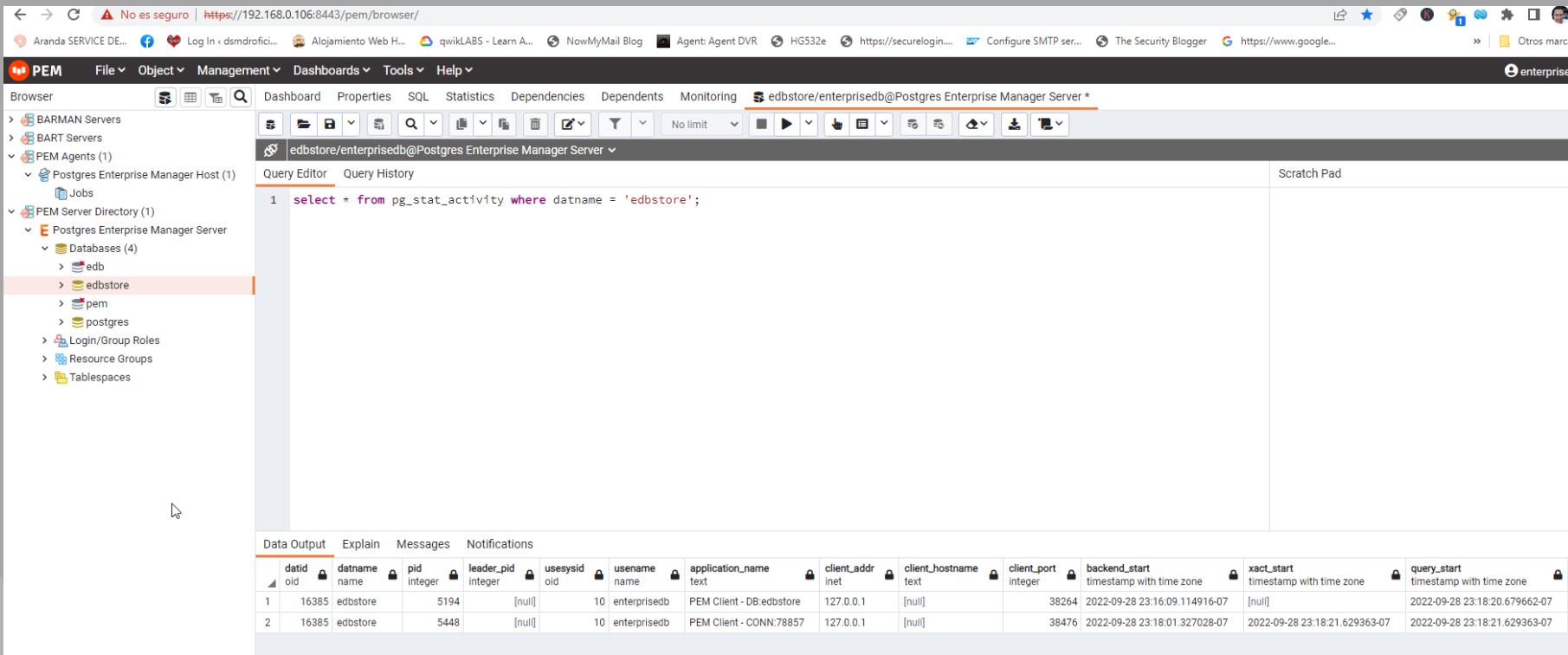
Instalación

- Escribe una query para ver la actual sesión en la base de datos.
- Escribe una query para ver cuantos usuarios son “idle” en la sesión.
- Escribe una query para ver cuantos usuarios están conectados con la base “edbstore”.
- Abre una terminal y conéctate a la base edbstore usnado psql.
- Encuentra el id del usuario conectado.

LABORATORIO 4.

Instalación

Escribe una query para ver la actual sesión en la base de datos.



The screenshot shows a web-based interface for managing PostgreSQL databases. The left sidebar lists various database components: BARMAN Servers, BART Servers, PEM Agents (1), PEM Server Directory (1), and Postgres Enterprise Manager Server. Under the Postgres Enterprise Manager Server, there are four databases: edb, edbstore (which is selected and highlighted in pink), pem, and postgres. The main panel contains a Query Editor with the following SQL query:

```
1 select * from pg_stat_activity where datname = 'edbstore';
```

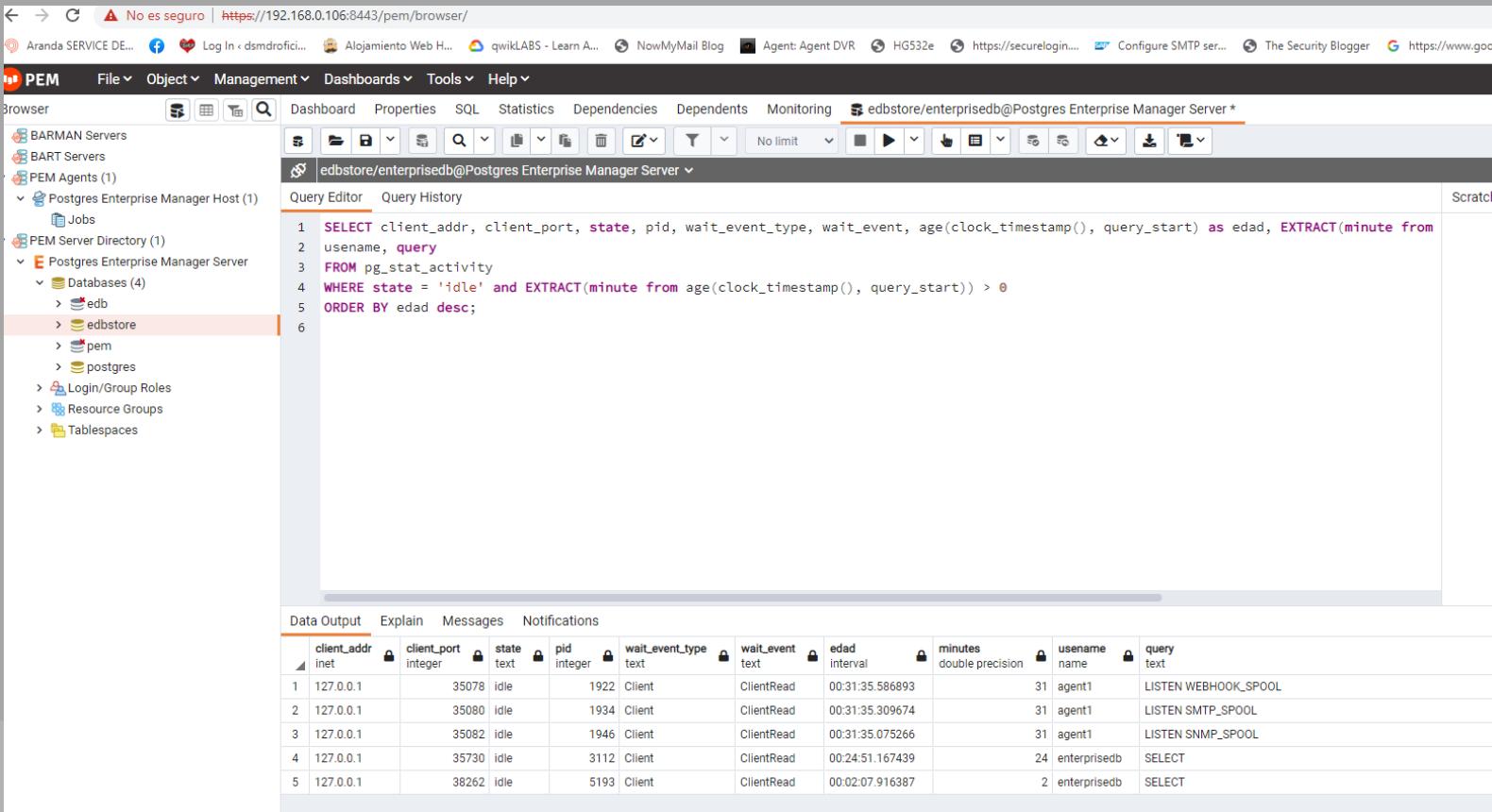
Below the query editor is a Data Output table showing two rows of session data. The columns include: datid, oid, dataname, pid, leader_pid, usesysid, username, application_name, client_addr, client_hostname, client_port, backend_start, xact_start, and query_start. The first row corresponds to the selected database 'edbstore' and the second row corresponds to 'pem'. The 'backend_start' column shows the timestamp of when each session began.

datid	oid	dataname	pid	leader_pid	usesysid	username	application_name	client_addr	client_hostname	client_port	backend_start	xact_start	query_start
1	16385	edbstore	5194	[null]	10	enterprisedb	PEM Client - DB.edbstore	127.0.0.1	[null]	38264	2022-09-28 23:16:09.114916-07	[null]	2022-09-28 23:18:20.679662-07
2	16385	edbstore	5448	[null]	10	enterprisedb	PEM Client - CONN:78857	127.0.0.1	[null]	38476	2022-09-28 23:18:01.327028-07	2022-09-28 23:18:21.629363-07	2022-09-28 23:18:21.629363-07

LABORATORIO 4.

Instalación

Escribe una query para ver la actual sesión en la base de datos.
Escribe una query para ver cuantos usuarios son “idle” en la sesión.



The screenshot shows the Postgres Enterprise Manager (PEM) interface. The left sidebar displays a tree view of database objects: BARMAN Servers, BART Servers, PEM Agents (1), Postgres Enterprise Manager Host (1) which contains Jobs, PEM Server Directory (1), Postgres Enterprise Manager Server (which has Databases: edb, edbstore, pem, postgres), Login/Group Roles, Resource Groups, and Tablespaces. The 'edbstore' database is selected. The main area has tabs for 'Query Editor' (selected) and 'Query History'. The Query Editor contains the following SQL code:

```
1 SELECT client_addr, client_port, state, pid, wait_event_type, wait_event, age(clock_timestamp(), query_start) as edad, EXTRACT(minute from
2 username, query
3 FROM pg_stat_activity
4 WHERE state = 'idle' and EXTRACT(minute from age(clock_timestamp(), query_start)) > 0
5 ORDER BY edad desc;
6
```

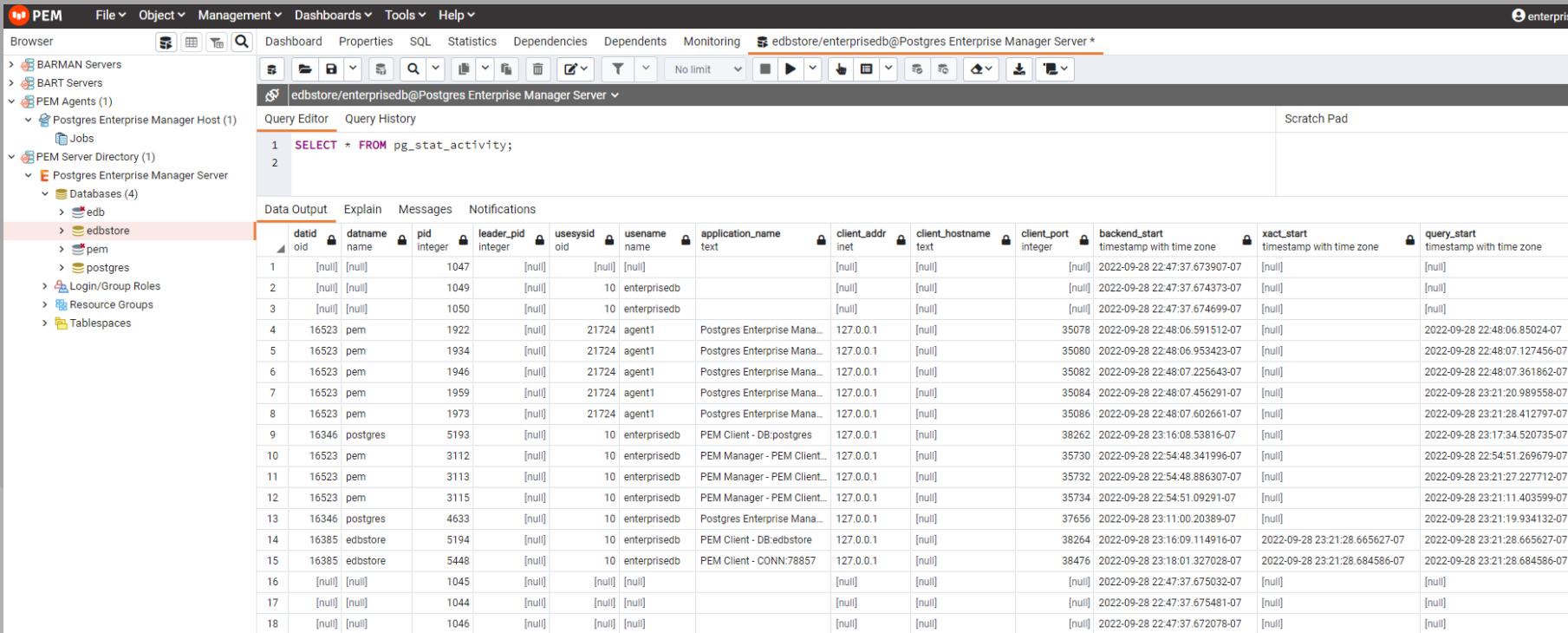
The Data Output tab shows the results of the query:

	client_addr	client_port	state	pid	wait_event_type	wait_event	edad	minutes	username	query
1	127.0.0.1	35078	idle	1922	Client	ClientRead	00:31:35.586893	31	agent1	LISTEN WEBHOOK_SPOOL
2	127.0.0.1	35080	idle	1934	Client	ClientRead	00:31:35.309674	31	agent1	LISTEN SMTP_SPOOL
3	127.0.0.1	35082	idle	1946	Client	ClientRead	00:31:35.075266	31	agent1	LISTEN SNMP_SPOOL
4	127.0.0.1	35730	idle	3112	Client	ClientRead	00:24:51.167439	24	enterprisedb	SELECT
5	127.0.0.1	38262	idle	5193	Client	ClientRead	00:02:07.916387	2	enterprisedb	SELECT

LABORATORIO 4.

Instalación

- Escribe una query para ver la actual sesión en la base de datos.
- Escribe una query para ver cuantos usuarios son “idle” en la sesión.
- Escribe una query para ver cuantos usuarios están conectados con la base “edbstore”.



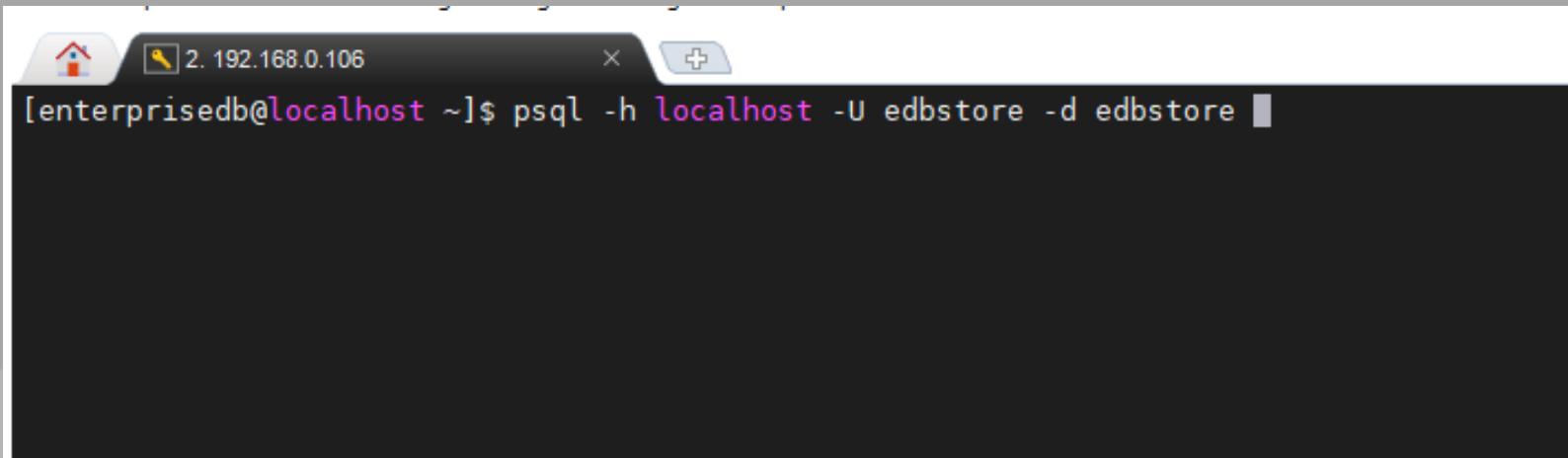
The screenshot shows the Postgres Enterprise Manager (PEM) interface. On the left, the navigation tree includes sections like Browser, PEM Agents, PEM Server Directory, and Databases. The Databases section is expanded, showing 'edbstore' as the selected database. In the center, the Query Editor tab is active, displaying the SQL query: 'SELECT * FROM pg_stat_activity;'. Below the query, the Data Output tab is selected, showing a table of activity statistics. The table has columns: datid, datname, pid, leader_pid, usesysid, username, application_name, client_addr, client_hostname, client_port, backend_start, xact_start, and query_start. The data shows multiple sessions, mostly for the 'edbstore' database, with various users and application names like 'agent1' and 'PEM Client - DB:edbstore'.

datid	datname	pid	leader_pid	usesysid	username	application_name	client_addr	client_hostname	client_port	backend_start	xact_start	query_start
old	name	integer	integer	oid	name	text	inet	text	integer	timestamp with time zone	timestamp with time zone	timestamp with time zone
1	[null]	[null]	1047	[null]	[null]		[null]	[null]	[null]	2022-09-28 22:47:37.673907-07	[null]	[null]
2	[null]	[null]	1049	[null]	10	enterprisedb	[null]	[null]	[null]	2022-09-28 22:47:37.674373-07	[null]	[null]
3	[null]	[null]	1050	[null]	10	enterprisedb	[null]	[null]	[null]	2022-09-28 22:47:37.674699-07	[null]	[null]
4	16523	pem	1922	[null]	21724	agent1	Postgres Enterprise Mana...	127.0.0.1	[null]	35078	2022-09-28 22:48:06.591512-07	[null]
5	16523	pem	1934	[null]	21724	agent1	Postgres Enterprise Mana...	127.0.0.1	[null]	35080	2022-09-28 22:48:06.953423-07	[null]
6	16523	pem	1946	[null]	21724	agent1	Postgres Enterprise Mana...	127.0.0.1	[null]	35082	2022-09-28 22:48:07.225643-07	[null]
7	16523	pem	1959	[null]	21724	agent1	Postgres Enterprise Mana...	127.0.0.1	[null]	35084	2022-09-28 22:48:07.456291-07	[null]
8	16523	pem	1973	[null]	21724	agent1	Postgres Enterprise Mana...	127.0.0.1	[null]	35086	2022-09-28 22:48:07.602661-07	[null]
9	16346	postgres	5193	[null]	10	enterprisedb	PEM Client - DB:postgres	127.0.0.1	[null]	38262	2022-09-28 23:16:08.53816-07	[null]
10	16523	pem	3112	[null]	10	enterprisedb	PEM Manager - PEM Client...	127.0.0.1	[null]	35730	2022-09-28 22:54:48.341996-07	[null]
11	16523	pem	3113	[null]	10	enterprisedb	PEM Manager - PEM Client...	127.0.0.1	[null]	35732	2022-09-28 22:54:48.886307-07	[null]
12	16523	pem	3115	[null]	10	enterprisedb	PEM Manager - PEM Client...	127.0.0.1	[null]	35734	2022-09-28 22:54:51.09291-07	[null]
13	16346	postgres	4633	[null]	10	enterprisedb	Postgres Enterprise Mana...	127.0.0.1	[null]	37656	2022-09-28 23:11:00.20389-07	[null]
14	16385	edbstore	5194	[null]	10	enterprisedb	PEM Client - DB:edbstore	127.0.0.1	[null]	38264	2022-09-28 23:16:09.114916-07	2022-09-28 23:21:28.665627-07
15	16385	edbstore	5448	[null]	10	enterprisedb	PEM Client - CONN:78857	127.0.0.1	[null]	38476	2022-09-28 23:18:01.327028-07	2022-09-28 23:21:28.684586-07
16	[null]	[null]	1045	[null]	[null]	[null]	[null]	[null]	[null]	2022-09-28 22:47:37.675032-07	[null]	[null]
17	[null]	[null]	1044	[null]	[null]	[null]	[null]	[null]	[null]	2022-09-28 22:47:37.675481-07	[null]	[null]
18	[null]	[null]	1046	[null]	[null]	[null]	[null]	[null]	[null]	2022-09-28 22:47:37.672078-07	[null]	[null]

LABORATORIO 4.

Instalación

- Escribe una query para ver la actual sesión en la base de datos.
- Escribe una query para ver cuantos usuarios son “idle” en la sesión.
- Escribe una query para ver cuantos usuarios están conectados con la base “edbstore”.
- Abre una terminal y conéctate a la base edbstore usnado psql:
 - [enterprisedb@localhost ~]\$ psql -h localhost -U edbstore -d edbstore



```
[enterprisedb@localhost ~]$ psql -h localhost -U edbstore -d edbstore
```

LABORATORIO 4.

Instalación

- Abre una terminal y conéctate a la base edbstore usnado psql.
- Encuentra el id del usuario conectado.

```
[enterprisedb@localhost ~]$ ps -ef | grep postgres
enterpr+ 1039  987  0 22:47 ?          00:00:00 postgres: logger
enterpr+ 1044  987  0 22:47 ?          00:00:00 postgres: checkpointer
enterpr+ 1045  987  0 22:47 ?          00:00:00 postgres: background writer
enterpr+ 1046  987  0 22:47 ?          00:00:00 postgres: walwriter
enterpr+ 1047  987  0 22:47 ?          00:00:00 postgres: autovacuum launcher
enterpr+ 1048  987  0 22:47 ?          00:00:00 postgres: stats collector
enterpr+ 1049  987  0 22:47 ?          00:00:00 postgres: dbms_aq launcher
enterpr+ 1050  987  0 22:47 ?          00:00:00 postgres: logical replication launcher
enterpr+ 1922  987  0 22:48 ?          00:00:00 postgres: agent1 pem 127.0.0.1(35078) idle
enterpr+ 1934  987  0 22:48 ?          00:00:00 postgres: agent1 pem 127.0.0.1(35080) idle
enterpr+ 1946  987  0 22:48 ?          00:00:00 postgres: agent1 pem 127.0.0.1(35082) idle
enterpr+ 1959  987  0 22:48 ?          00:00:07 postgres: agent1 pem 127.0.0.1(35084) idle
enterpr+ 1973  987  4 22:48 ?          00:01:33 postgres: agent1 pem 127.0.0.1(35086) idle
enterpr+ 3112  987  0 22:54 ?          00:00:00 postgres: enterprisedb pem 127.0.0.1(35730) idle
enterpr+ 3113  987  0 22:54 ?          00:00:03 postgres: enterprisedb pem 127.0.0.1(35732) idle
enterpr+ 3115  987  0 22:54 ?          00:00:03 postgres: enterprisedb pem 127.0.0.1(35734) idle
enterpr+ 4633  987  0 23:10 ?          00:00:00 postgres: enterprisedb postgres 127.0.0.1(37656) idle
enterpr+ 5193  987  0 23:16 ?          00:00:00 postgres: enterprisedb postgres 127.0.0.1(38262) idle
enterpr+ 5194  987  0 23:16 ?          00:00:03 postgres: enterprisedb edbstore 127.0.0.1(38264) idle
enterpr+ 5448  987  0 23:18 ?          00:00:00 postgres: enterprisedb edbstore 127.0.0.1(38476) idle
enterpr+ 6085  4871  0 23:22 pts/0    00:00:00 grep --color=auto postgres
[enterprisedb@localhost ~]$ █
```

SQL TUNING.

- Checa sintaxis
- Llama tráfico
- Identifica tipo de Query
- Comando Predesesor
- Querys Rotas

PARSEO

- Statement Processing
- Se encarga de:

Optimiza

- Plan de Ejecución
- Estadísticas
- Optimizadores
- Costo de Querys
- Mejor Plan de Ejecución

- Ejecuta Query basada en un plan

Executa

SQL TUNING.

Errores Comunes en Querys

¿Quién no ha escuchado la siguiente afirmación?

“la Base de Datos va muy lenta”

➤ Reglas generales para afrontar problemas de rendimiento:

- La mayoría de los problemas de rendimiento no suelen estar en la Base de Datos.
- Menos del 10% de los problemas de rendimiento de tu sistema son los causantes de una reducción del rendimiento de aproximadamente del 90%.
- En todo momento y normalmente, solo es posible observar e identificar el problema cuando esta sucediendo.
- Las aplicaciones en más del 80% y Querys internas son en la mayoría de los casos los culpables.

SQL TUNING.

Errores Comunes en Querys

- Full table scans.
- Bad SQL.
- Sorts using disk.
- Join orders.
- Old or mis.sing statistics.
- I/O issues.
- Bad connection management.

SQL TUNING.

Objetivos del Tuning.

- Identificar Querys Lentas.
- Encontrar la query problemática.
- Reducir los tiempos de ejecución.
- Reducir consumo de recursos de una query.
- Analizar plan de ejecución.
- Ver el workload balanceo.

SQL TUNING.

Podemos seguir el siguiente flujo de trabajo a la hora de afrontar un problema de rendimiento en el sistema:

1. Identify Slow SQL.
2. Review Explain Plan.
3. Optimizer Statistics and Behavior.
4. Restructure SQL Statements.
5. Add/Remove Indexes.
6. Review Final Execution Plan.

SQL TUNING.

Slow Query tracking Options.

- Ver duración de una consulta SQL.
- Usar PEM's Log Manager para configurar logs de querys lentas.
- Usar PEM's Postgres Log Analysis Expert Wizard para analizar los mensajes.
- Correr SQL Profiler trace para encontrar errores.
- Optimizar SQL Querys (La experiencia de un DBA es vital).

SQL TUNING.

Slow Query Parameter.

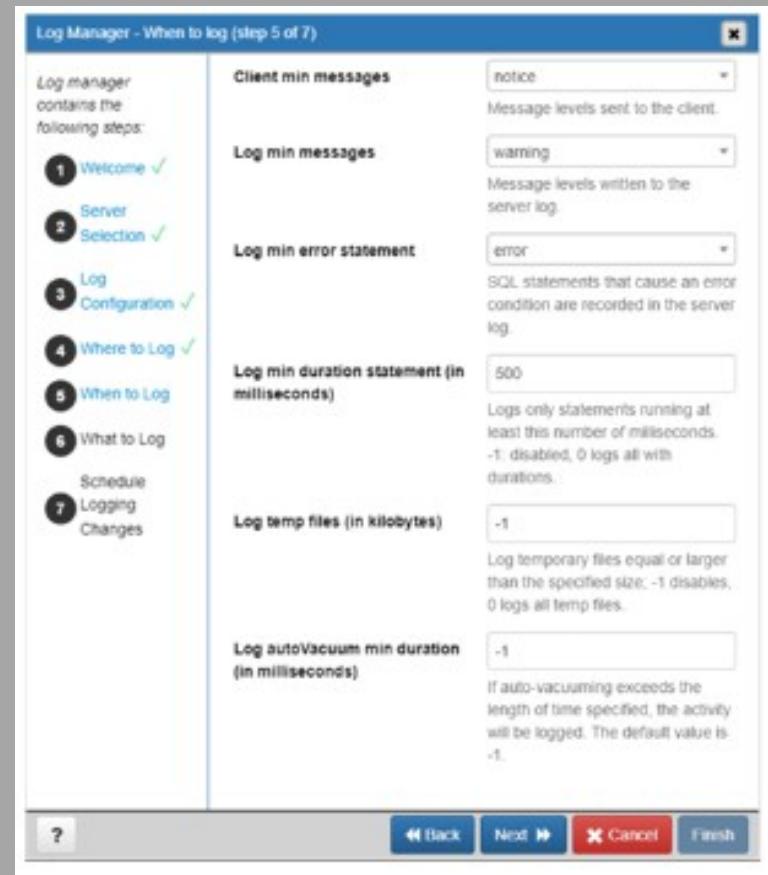
1. `log_min_duration_statement`

- Setealo en milisegundos.
- Todas las consultas SQL que corran en el tiempo marcado serán grabadas.
- Habilitar esta opción para identificar issues en las aplicaciones.

SQL TUNING.

PEM Log Manager.

- La instancia debe de ser registrada como PEM managed server.
- Abre LOG Manager desde el menú principal de PEM.
- Configura Log Min Duration Statement con el wizard o asistente.
- Usa el dashboard Server Log Análisis.
- Si es necesario utiliza el asistente de análisis de PEM.



SQL TUNING.

PEM Log Analysis Expert.

Reporte de Ejemplo

Summary Statistics

Statistics	Values
Total queries	375
Total queries duration	00:00:53.639328
First query	02/03/2018 22:41:37.183 MST
Last query	03/03/2018 22:19:30.246 MST
Queries peak time	03/03/2018 16:24:26 MST queries 4
Number of events	401
Number of unique events	1
Total number of sessions	376
Total duration of sessions	None
Average sessions duration	None
Total number of connections	0
Total number of databases	0

Hourly DML Statistics

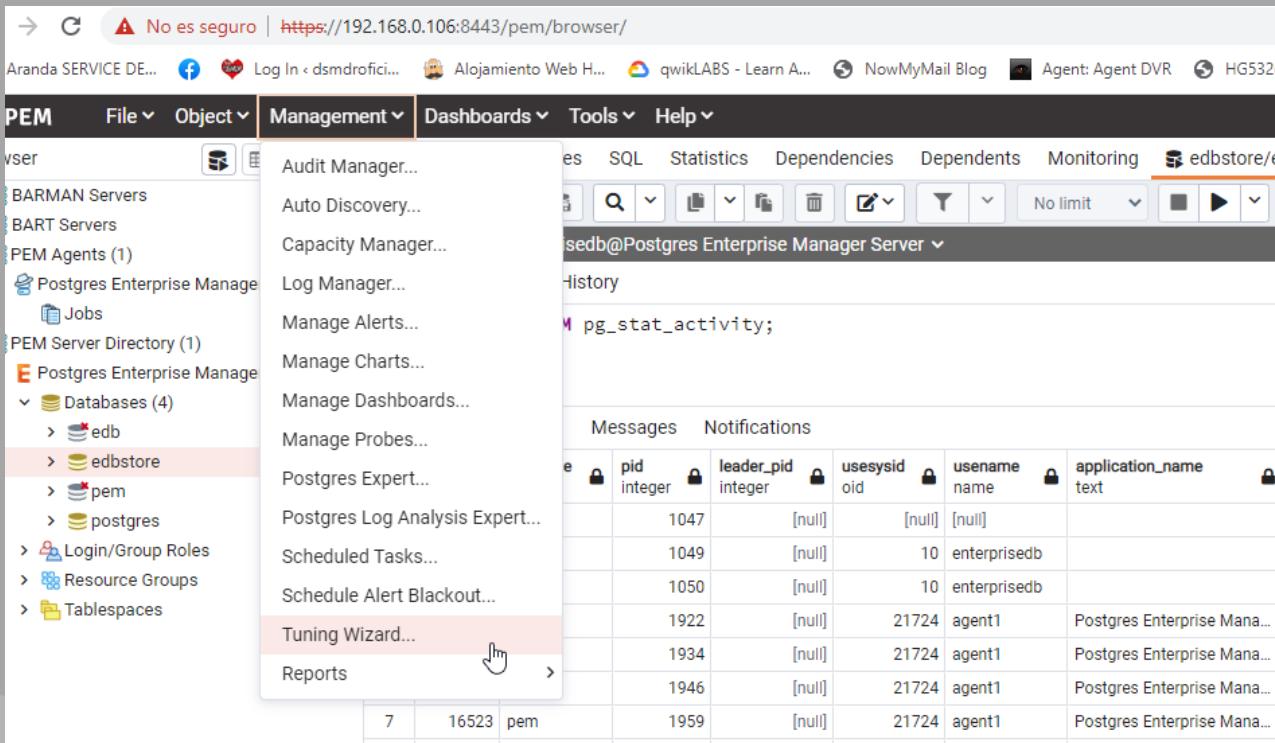
Time	Database	Command Type	Total Count	Min Duration (ms)	Max Duration (ms)	Avg Duration (ms)
02/03/2018 22:00	edb	SELECT	9	101.08	221.86	142.86
02/03/2018 22:00	postgres	SELECT	6	107.70	219.42	149.26
02/03/2018 22:00	pred	SELECT	5	101.03	149.35	123.58
02/03/2018 22:00	pred	UPDATE	1	113.93	113.93	113.93
02/03/2018 23:00	edb	SELECT	12	113.60	139.95	133.92
02/03/2018 23:00	postgres	SELECT	2	104.95	104.95	104.95
02/03/2018 23:00	pred	SELECT	12	106.23	135.91	123.00
03/03/2018 00:00	edb	SELECT	7	127.16	135.72	132.16
03/03/2018 00:00	pred	SELECT	7	125.18	145.67	131.91
03/03/2018 01:00	edb	SELECT	6	117.82	138.40	130.97



SQL TUNING.

PEM Creating a New SQL Trace.

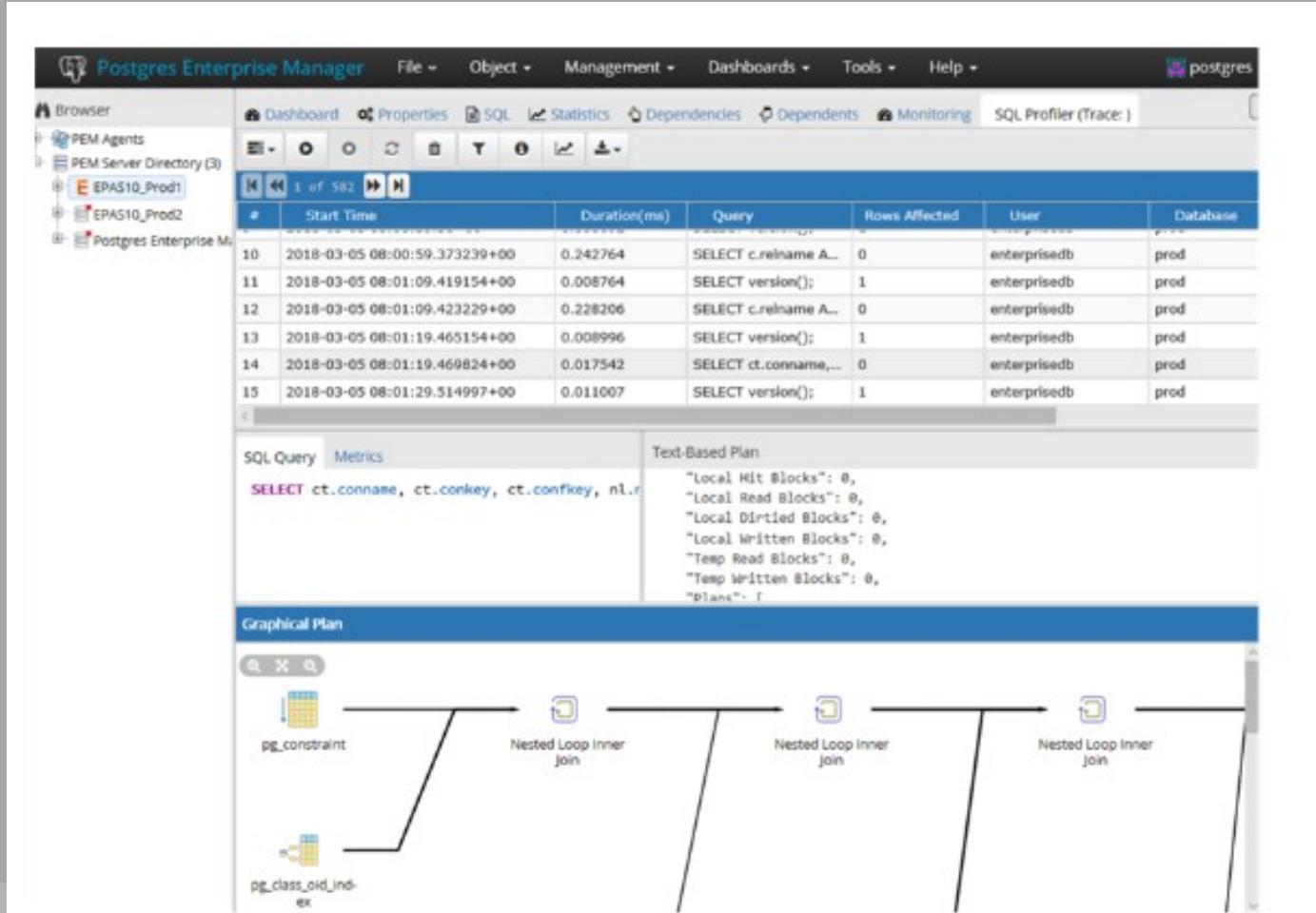
- Captura Workloads.
- Ve las características de las querys.
- Usa las recomendaciones de indexación.



The screenshot shows the Postgres Enterprise Manager (PEM) web interface. The URL is https://192.168.0.106:8443/pem/browser/. The navigation bar includes links for Aranda SERVICE DE..., Log In, Alojamiento Web H..., qwikLABS - Learn A..., NowMyMail Blog, Agent: Agent DVR, and HG532e. The main menu has tabs for PEM, File, Object, Management, Dashboards, Tools, and Help. The Management tab is selected. On the left, a sidebar lists various management components: user, BARMAN Servers, BART Servers, PEM Agents (1), Postgres Enterprise Manager (Jobs), PEM Server Directory (1), Postgres Enterprise Manager (Databases 4), Login/Group Roles, Resource Groups, and Tablespaces. The 'edbstore' database is currently selected. The main pane displays a table titled 'pg_stat_activity'. The table has columns: id, pid, leader_pid, usesysid, username, and application_name. The data shows several rows for different processes, all associated with the 'agent1' user and the 'Postgres Enterprise Manager' application. A 'Tuning Wizard...' button is visible at the bottom of the table area. The bottom of the screen shows a footer with page numbers 7 and 16523, and a 'pem' link.

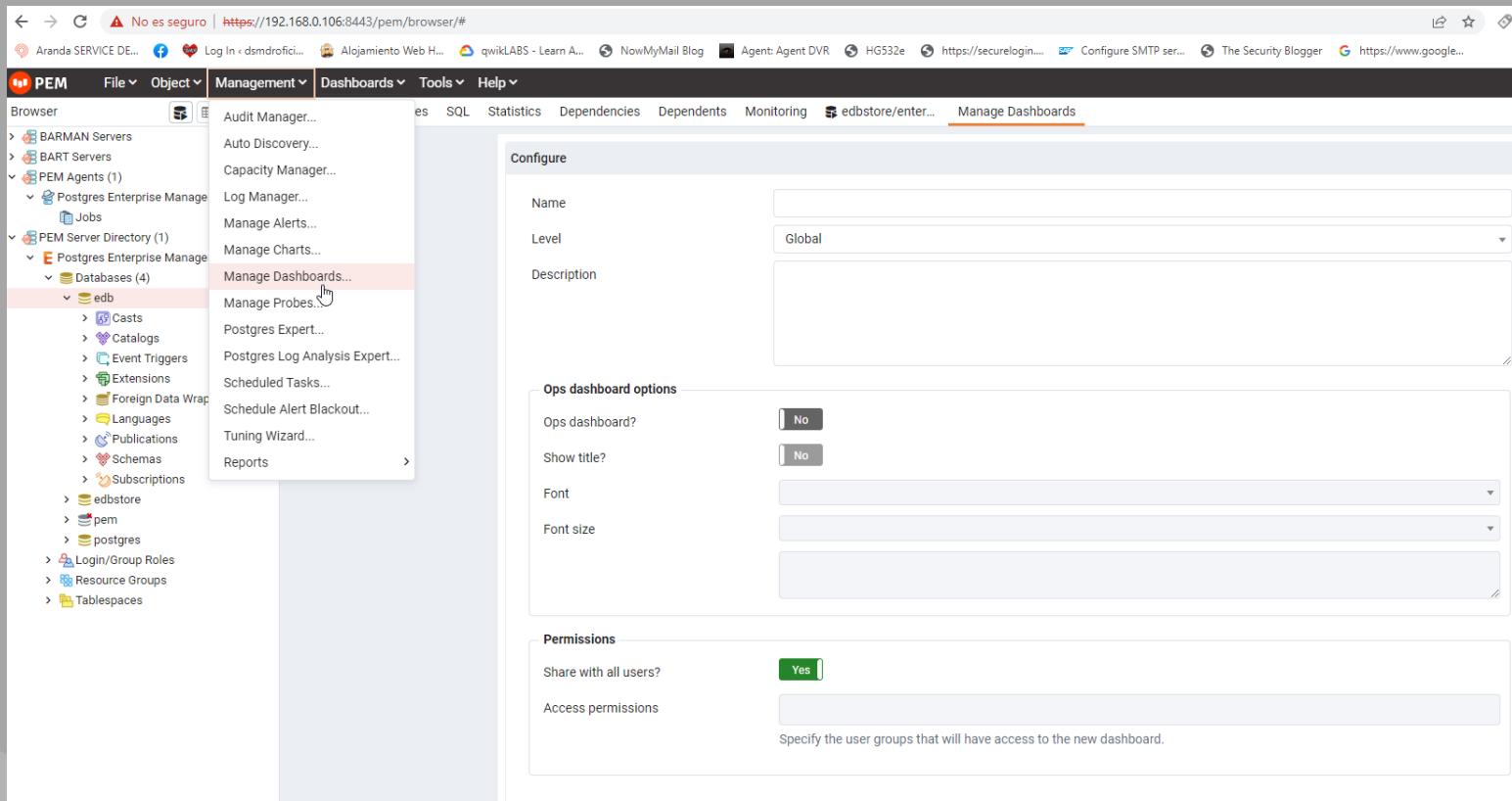
SQL TUNING.

Viewing the SQL Profiler Report.



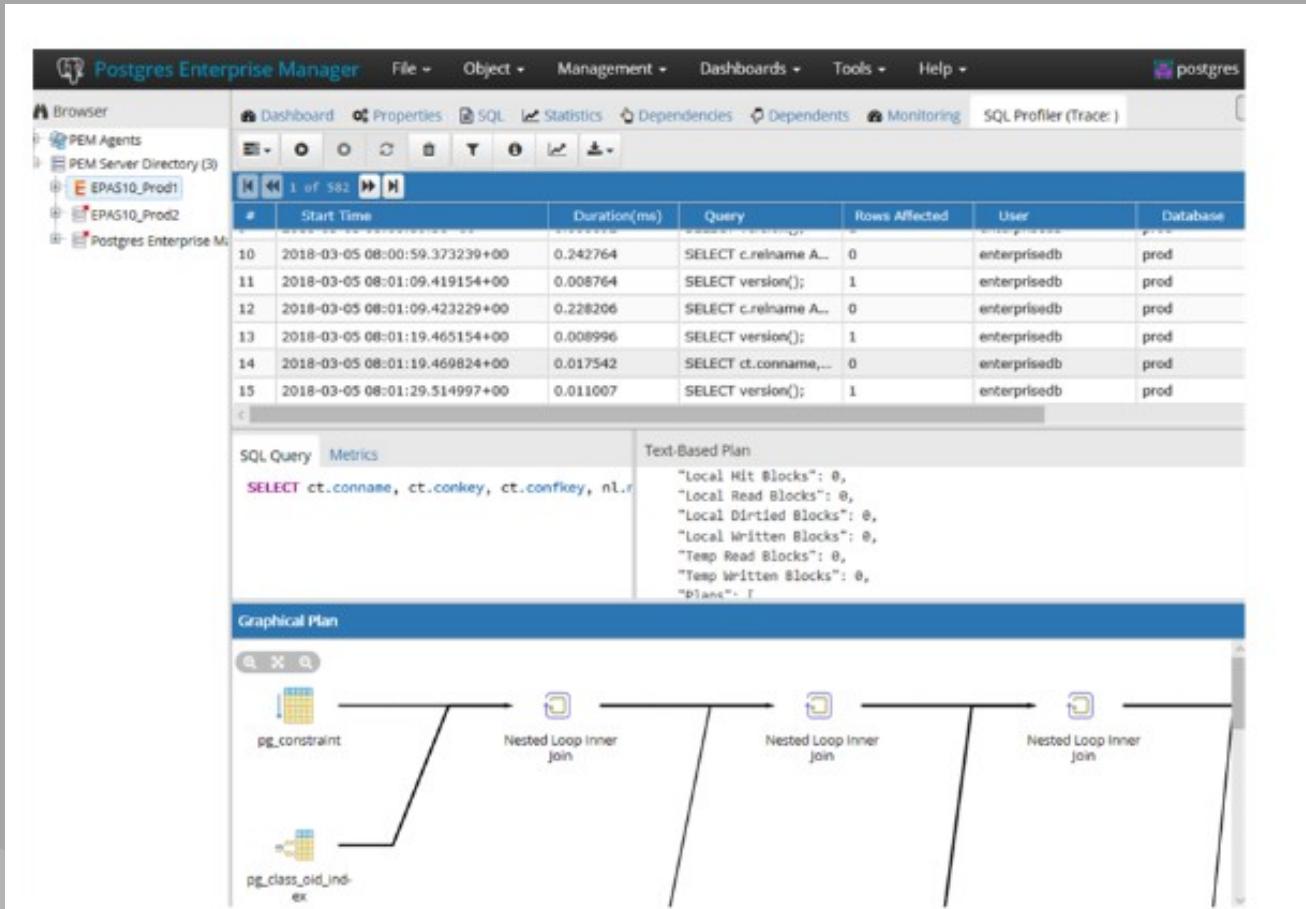
The screenshot shows the Postgres Enterprise Manager interface. The main window displays a SQL Profiler report with 15 rows of data. The columns are: #, Start Time, Duration(ms), Query, Rows Affected, User, and Database. The queries listed are mostly SELECT statements involving version() and column names like ct.conname, ct.conkey, and pg_constraint. The duration for most queries is very low (e.g., 0.008764 ms). The user is 'enterprisedb' and the database is 'prod'. Below the report, there are tabs for 'SQL Query' and 'Metrics'. The 'SQL Query' tab shows the query: `SELECT ct.conname, ct.conkey, ct.confkey, nlr...`. The 'Metrics' tab displays performance metrics: "Local Hit Blocks": 0, "Local Read Blocks": 0, "Local Dirtyed Blocks": 0, "Local Written Blocks": 0, "Temp Read Blocks": 0, "Temp Written Blocks": 0, and "Plan": 1. At the bottom, a 'Graphical Plan' section shows a query execution tree. It starts with a 'pg_constraint' node at the top level, which has three 'Nested Loop Inner join' children. These children point to a 'pg_class_oid_index' node at the bottom level. The nodes are represented by small icons: a grid for pg_constraint and a square with a circle for pg_class_oid_index.

- Crea un nuevo Dashboard usando PEM.
- Comenta los resultados.



SQL TUNING.

Viewing the SQL Profiler Report.



The screenshot displays the Postgres Enterprise Manager interface, specifically the SQL Profiler (Trace) tab. The main pane shows a table of 582 trace entries, with the first 15 rows listed below:

#	Start Time	Duration(ms)	Query	Rows Affected	User	Database
10	2018-03-05 08:00:59.373239+00	0.242764	SELECT c.relname A...	0	enterprisedb	prod
11	2018-03-05 08:01:09.419154+00	0.008764	SELECT version();	1	enterprisedb	prod
12	2018-03-05 08:01:09.423229+00	0.228206	SELECT c.relname A...	0	enterprisedb	prod
13	2018-03-05 08:01:19.465154+00	0.008996	SELECT version();	1	enterprisedb	prod
14	2018-03-05 08:01:19.469824+00	0.017542	SELECT ct.conname,...	0	enterprisedb	prod
15	2018-03-05 08:01:29.514997+00	0.011007	SELECT version();	1	enterprisedb	prod

Below the table, the SQL Query tab shows the query being analyzed:

```
SELECT ct.conname, ct.conkey, ct.confkey, nlr...
```

The Metrics tab displays performance metrics for the query, including:

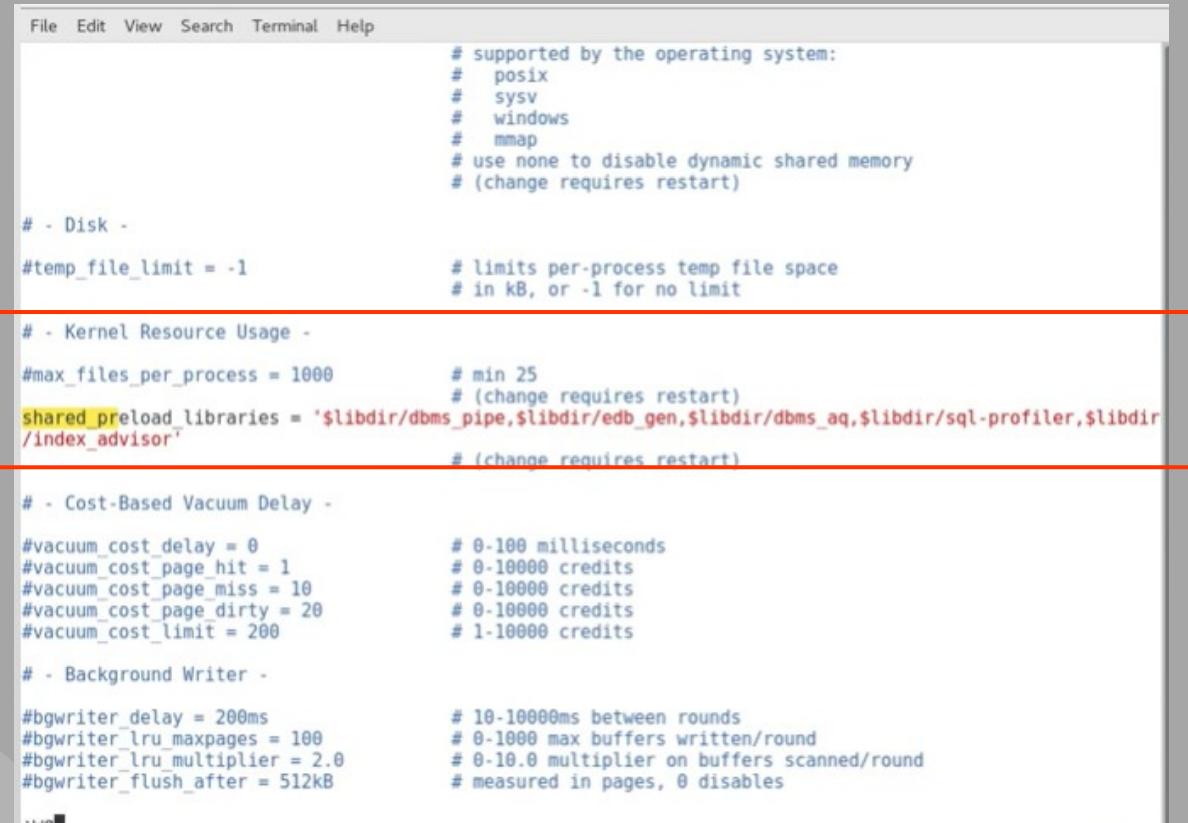
- "Local Hit Blocks": 0
- "Local Read Blocks": 0
- "Local Dirtyd Blocks": 0
- "Local Written Blocks": 0
- "Temp Read Blocks": 0
- "Temp Written Blocks": 0
- "Plans": 1

The Graphical Plan tab shows the execution plan as a tree diagram:

```
graph LR; pg_constraint --> NestedLoopInnerJoin1[Nested Loop Inner join]; pg_class_oid_index --> NestedLoopInnerJoin1; NestedLoopInnerJoin1 --> NestedLoopInnerJoin2[Nested Loop Inner join]; NestedLoopInnerJoin2 --> NestedLoopInnerJoin3[Nested Loop Inner join]; NestedLoopInnerJoin3 --> output
```

The plan consists of three nested loops. The innermost loop involves the `pg_constraint` table. The middle loop involves the `pg_class_oid_index` table. The outermost loop is a nested loop inner join between the two intermediate results.

- Agrega la siguiente línea al pg_hba.conf.
- Reinicia el servidor.
Con esto Activamos el SQL Profiles (viene por default apagado).
- Adicional se necesita correr el sql_profiler.sql (/usr/edb/as13/share/contrib/sql-profiler.sql).



```
File Edit View Search Terminal Help

# supported by the operating system:
# posix
# sysv
# windows
# mmap
# use none to disable dynamic shared memory
# (change requires restart)

# - Disk -
#temp_file_limit = -1          # limits per-process temp file space
# in kB, or -1 for no limit

# - Kernel Resource Usage -
#max_files_per_process = 1000      # min 25
# (change requires restart)
shared_preload_libraries = '$libdir/dbms_pipe,$libdir/edb_gen,$libdir/dbms_aq,$libdir/sql-profiler,$libdir/index_advisor'
# (change requires restart)

# - Cost-Based Vacuum Delay -
#vacuum_cost_delay = 0           # 0-100 milliseconds
#vacuum_cost_page_hit = 1         # 0-10000 credits
#vacuum_cost_page_miss = 10        # 0-10000 credits
#vacuum_cost_page_dirty = 20       # 0-10000 credits
#vacuum_cost_limit = 200          # 1-10000 credits

# - Background Writer -
#bgwriter_delay = 200ms          # 10-10000ms between rounds
#bgwriter_lru_maxpages = 100        # 0-1000 max buffers written/round
#bgwriter_lru_multiplier = 2.0       # 0-10.0 multiplier on buffers scanned/round
#bgwriter_flush_after = 512kB        # measured in pages, 0 disables
```

- Agrega la siguiente línea al pg_hba.conf.
 - Reinicia el servidor.
Con esto Activamos el SQL Profiles (viene por default apagado).
 - Adicional se necesita correr el sql_profiler.sql (/usr/edb/as13/share/contrib/sql-profiler.sql).

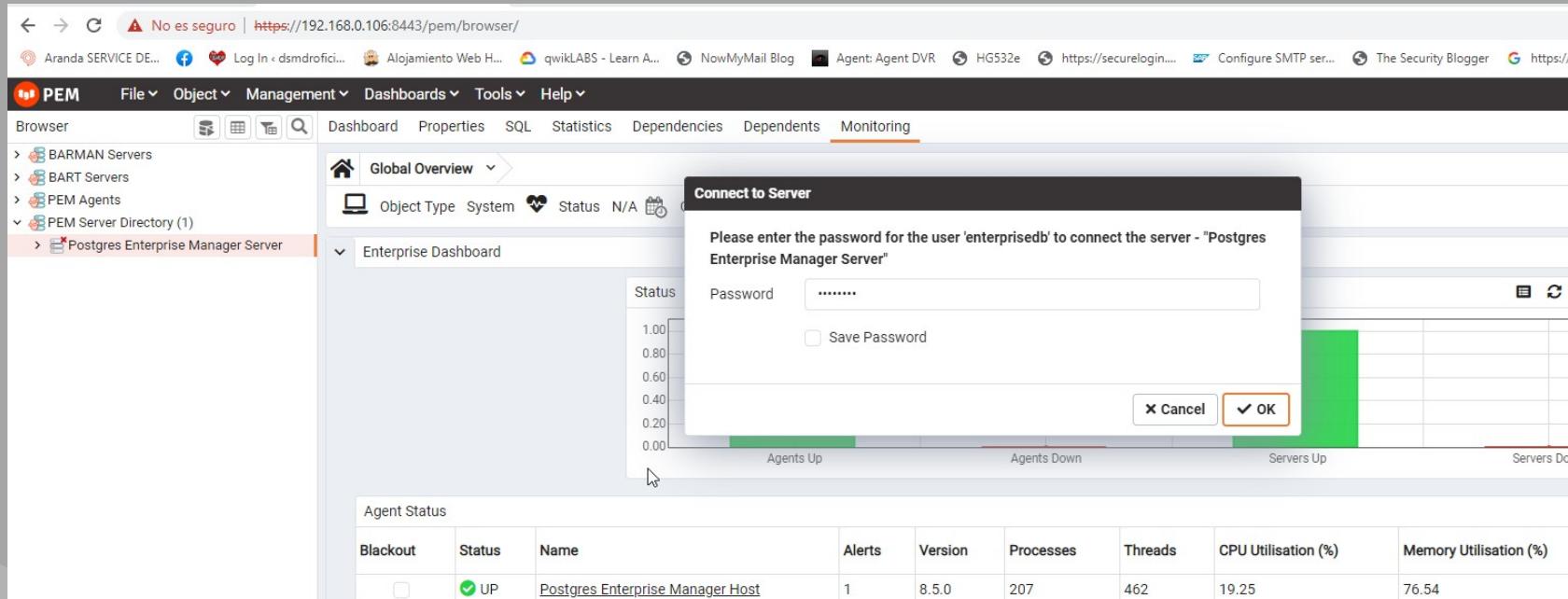
- Agrega la siguiente línea al pg_hba.conf.
- Reinicia el servidor.
Con esto Activamos el SQL Profiles (viene por default apagado).
- Adicional se necesita correr el sql_profiler.sql (/usr/edb/as13/share/index_advisor.sql).

```
[enterprisedb@localhost opt]$ psql -f /usr/edb/as13/share/contrib/index_advisor.sql edb
CREATE TABLE
CREATE INDEX
CREATE INDEX
CREATE FUNCTION
CREATE FUNCTION
CREATE VIEW
[enterprisedb@localhost opt]$ █
```

LABORATORIO

Instalación

- Regresamos a PEM.
- Damos LOGOUT.
- Damos LOGIN.



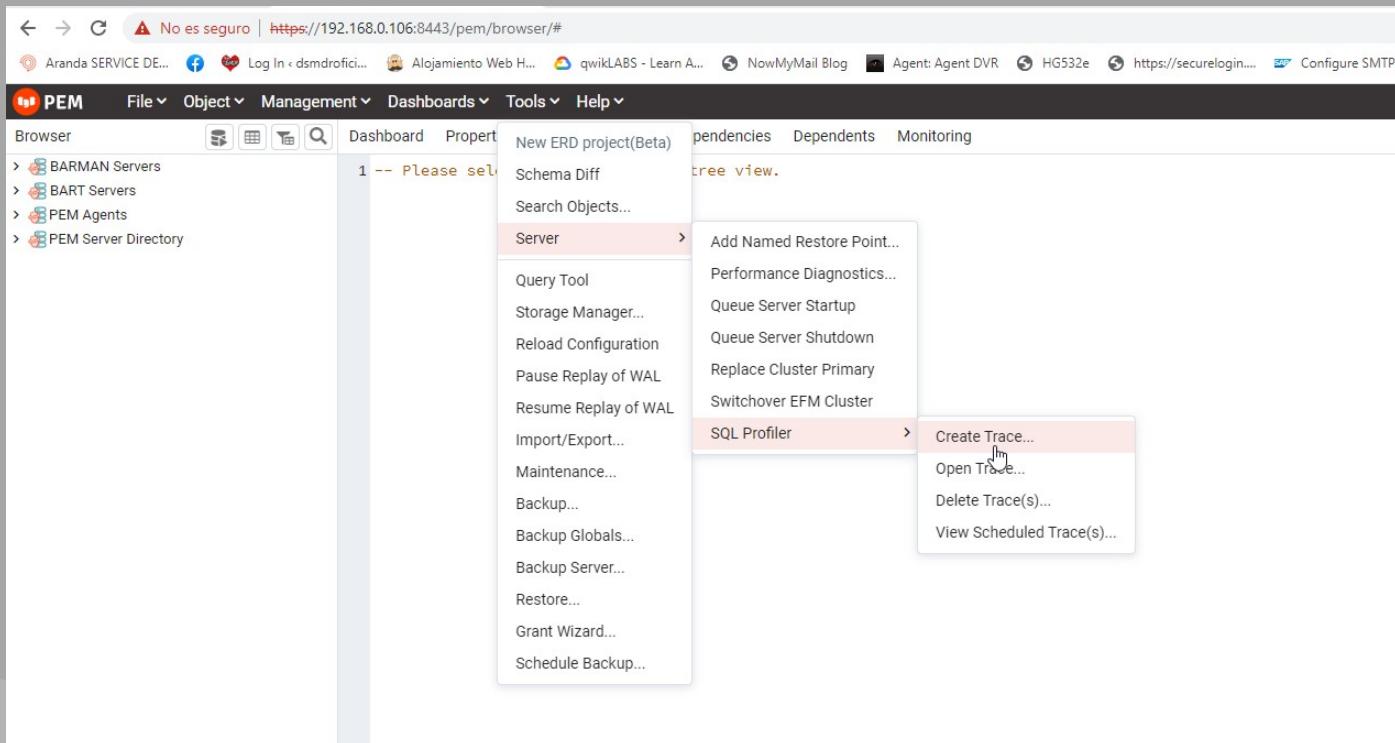
The screenshot shows the PEM web interface at <https://192.168.0.106:8443/pem/browser/>. The navigation bar includes File, Object, Management, Dashboards, Tools, and Help. The Monitoring tab is selected. The left sidebar lists BARMAN Servers, BART Servers, PEM Agents, and PEM Server Directory (1), with Postgres Enterprise Manager Server selected. The main area displays a 'Global Overview' dashboard with sections for Object Type, System, Status, and N/A. A 'Connect to Server' dialog box is overlaid, prompting for the password for the user 'enterprisedb'. The dialog contains fields for 'Password' (with a masked value), a 'Save Password' checkbox, and 'OK' and 'Cancel' buttons. Below the dialog, there is a chart showing Agent Status (Agents Up, Agents Down, Servers Up, Servers Down) and an 'Agent Status' table.

Blackout	Status	Name	Alerts	Version	Processes	Threads	CPU Utilisation (%)	Memory Utilisation (%)
<input type="checkbox"/>	 UP	Postgres Enterprise Manager Host	1	8.5.0	207	462	19.25	76.54

LABORATORIO

Instalación

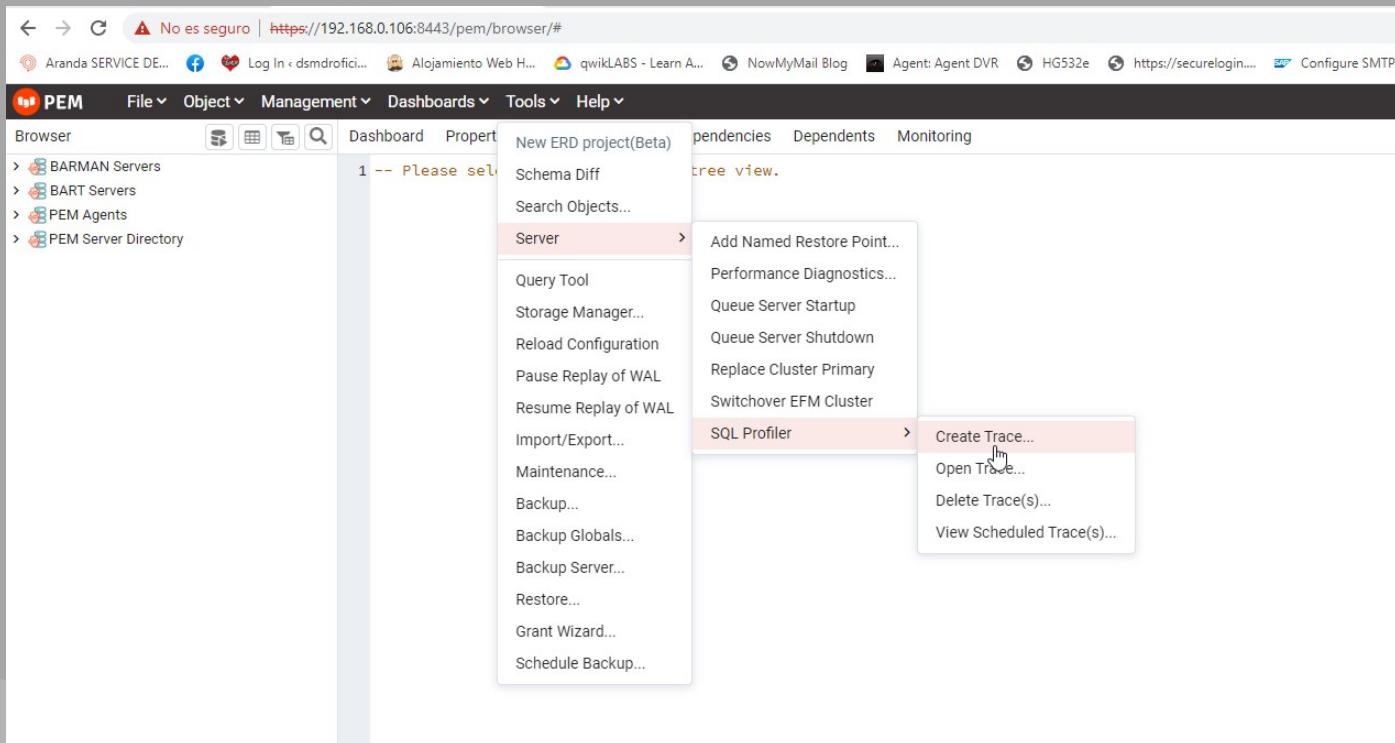
- Regresamos a PEM.
- Damos LOGOUT.
- Damos LOGIN.



LABORATORIO

Instalación

- Regresamos a PEM.
- Damos LOGOUT.
- Damos LOGIN.



SQL TUNING.

Tracking Execution Statistics.

- Tracking a query usando:
 - pg_stat statements extension
- Estadísticas en todo el cluster.
 - pg_stat_statements
- Vistas y parámetros.
 - pg_stat_statements extension
- Ver estadísticas de una Query.
 - View- pg_stat_statements
- Se utiliza para resetear las estadísticas.
 - Function - pg_stat_statements_reset

SQL TUNING.

Pg_stat_statements SETUP.

- Agrega pg_stat_statements a las librerias en postgresql.conf
- Configura parametros:
 - - pg_stat_statements.max - Maximo numero de tracks statements, default 5000.
 - - pg_stat_statements.track Default is top.
 - - pg_stat_statements.track utility- Track commands.

SELECT, INSERT, UPDATE and DELETE. Default is on.

- - pg_stat_statements.save - Guardar ante un server shutdowns. Default es ON.
- Reinicia la Database Cluster.
- Conecta ala database y habilita pg_stat_statements extension.

SQL TUNING.

Componentes de un plan de Ejecución.

- Un plan de ejecución muestra los pasos detallados necesario para ejecutar una sentencia SQL.
- Planner es responsable de generar la ejecución plan.
- El Optimizer determina la ejecución más eficiente plan.
- La optimización se basa en costos, el costo es un recurso estimado uso para un plan.
- El comando EXPLAIN se usa para ver un plan de consulta.

SQL TUNING.

Componentes de un plan de Ejecución.

- Cardinalidad - Row Estimates.
- Método de Acceso - Sequential or Index.
- Join Method - Hash, Nested Loop etc.
- Join Type, Join Order.
- Sort and Aggregates.

Syntax:

```
=# EXPLAIN [ ( option [, ...]1 ) 1 statement
EXPLAIN [ ANALYZE ] [ VERBOSE ] statement
where option can be one of:
ANALYZE [ boolean
VERBOSE [ boolean
COSTS [ boolean
BUFFERS [ boolean
TIMING [ boolean ]
SUMMARY [ boolean
FORMAT { TEXT XML JSON YAML }
```

SQL TUNING.

Ejemplo Múltiples Tablas.

```
edb=# EXPLAIN SELECT * FROM emp;
          QUERY PLAN
-----
Seq Scan on emp  (cost=0.00..1.14 rows=14
width=135)
```

Los números citados por EXPLAIN son:

- Costo estimado de puesta en marcha.
- Costo total estimado.
- Número estimado de salida de filas por este nodo del plan.
- Ancho promedio estimado (en bytes) de las filas generadas por este nodo del plan.

SQL TUNING.

Ejemplo Múltiples Tablas, Carga y Análisis.

Carga datos:

- =# INSERT INTO city
VALUES (1, 'Edmonton'), (2, 'Calgary'), (3, 'Sherwood Park'), (4, 'ST Albert');
- =# INSERT INTO office VALUES
(generate_series(1,100),4);
- =# INSERT INTO office VALUES
(generate_series(101,200),3);
- =# INSERT INTO office VALUES
(generate_series(201,300),2);
- =# INSERT INTO office VALUES
(generate_series(301,400),1);

Salida:

- =# ANALYZE citys
- =# ANALYZE office;

SQL TUNING.

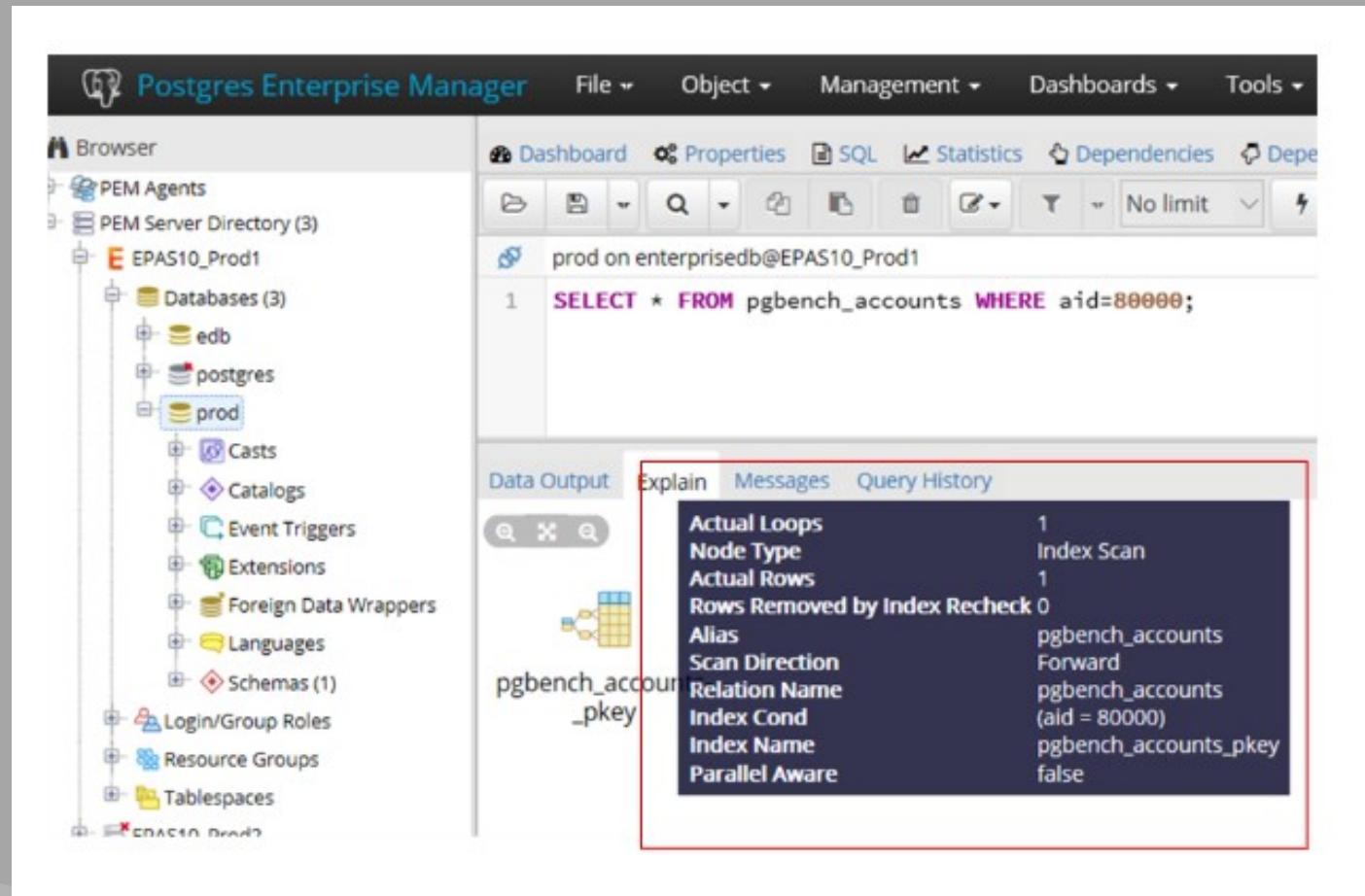
Análisis del plan.

- Plan:

```
=# EXPLAIN ANALYZE SELECT city.cityname, office.officeid,
office.cityid
    FROM city, office WHERE office.cityid = city.cityid;
Hash Join  (cost=1.09..12.59 rows=400 width=20) (actual
time=0.057..0.770 rows=400 loops=1)
  Hash Cond: (office.cityid = city.cityid)
    -> Seq Scan on office  (cost=0.00..6.00 rows=400
width=10) (actual time=0.012..0.128 rows=400 loops=1)
        -> Hash  (cost=1.04..1.04 rows=4 width=15) (actual
time=0.013..0.013 rows=4 loops=1)
          Buckets: 1024  Batches: 1  Memory Usage: 1kB
            -> Seq Scan on city  (cost=0.00..1.04 rows=4
width=15) (actual time=0.002..0.005 rows=4 loops=1)
Planning time: 0.577 ms
Execution time: 0.893 ms
```

SQL TUNING.

PEM QUERY TOOL.



The screenshot shows the Postgres Enterprise Manager (PEM) interface. On the left, the Browser pane displays the database structure under 'EPAS10_Prod1'. In the center, the main workspace shows a query in the SQL tab:

```
prod on enterpriseedb@EPAS10_Prod1
1 SELECT * FROM pgbench_accounts WHERE aid=80000;
```

Below the query, the Explain tab is selected, displaying the execution plan for the query. The plan details the following parameters:

Actual Loops	1
Node Type	Index Scan
Actual Rows	1
Rows Removed by Index Recheck	0
Alias	pgbench_accounts
Scan Direction	Forward
Relation Name	pgbench_accounts
Index Cond	(aid = 80000)
Index Name	pgbench_accounts_pkey
Parallel Aware	false

SQL TUNING.

EJEMPLO PSQL Tuning.

```
[enterprisedb@vm1 ~]$ psql edbstore edbuser
Password for user edbuser:
psql.bin (10.1.5)
Type "help" for help.

edbstore=> explain select * from customers;
               QUERY PLAN
-----
 Seq Scan on customers  (cost=0.00..706.00 rows=20000 width=278)
(1 row)

edbstore=> select relpages,reltuples from pg_class where relname='customers';
   relpages |   reltuples
-----+-----
      506 |     20000
(1 row)

edbstore=> show seq_page_cost ;
 seq_page_cost
-----
 1
(1 row)

edbstore=> show cpu_tuple_cost ;
 cpu_tuple_cost
-----
 0.01
(1 row)

edbstore=> select 506*1+20000*0.01;
 ?column?
-----
 706.00
(1 row)

edbstore=>
```

SQL TUNING.

EJEMPLO PSQL Tuning.

```
[enterprisedb@vml ~]$ psql edbstore edbuser
Password for user edbuser:
psql.bin (10.1.5)
Type "help" for help.

edbstore=> explain select * from customers;
               QUERY PLAN
-----
 Seq Scan on customers  (cost=0.00..706.00 rows=20000 width=278)
(1 row)

edbstore=> select relpages,reltuples from pg_class where relname='customers';
   relpages |   reltuples
-----+-----
      506 |     20000
(1 row)

edbstore=> show seq_page_cost ;
 seq_page_cost
-----
 1
(1 row)

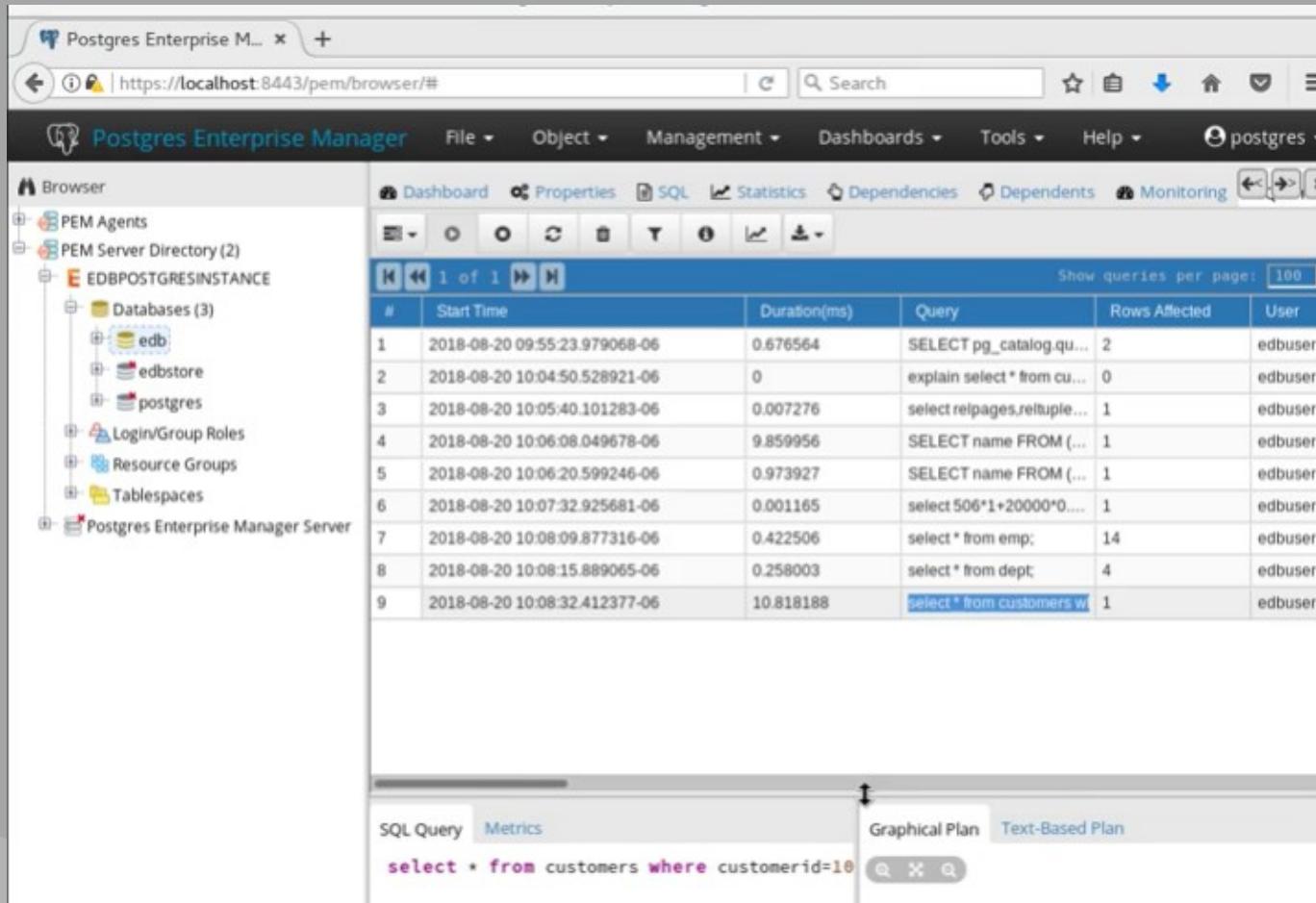
edbstore=> show cpu_tuple_cost ;
 cpu_tuple_cost
-----
 0.01
(1 row)

edbstore=> select 506*1+20000*0.01;
 ?column?
-----
 706.00
(1 row)

edbstore=>
```

SQL TUNING.

Se analizan los resultados PEM.



The screenshot shows the Postgres Enterprise Manager (PEM) interface. On the left, the 'Browser' pane displays a tree structure of database objects under 'EDBPOSTGRESINSTANCE'. The 'Databases' node is expanded, showing 'edb', 'edbstore', and 'postgres'. Other nodes include 'Login/Group Roles', 'Resource Groups', 'Tablespaces', and 'Postgres Enterprise Manager Server'. The main pane is titled 'EDBPOSTGRESINSTANCE' and contains a table of recent queries. The table has columns: #, Start Time, Duration(ms), Query, Rows Affected, and User. The data is as follows:

#	Start Time	Duration(ms)	Query	Rows Affected	User
1	2018-08-20 09:55:23.979068-06	0.676564	SELECT pg_catalog.qu...	2	edbuser
2	2018-08-20 10:04:50.528921-06	0	explain select * from cu...	0	edbuser
3	2018-08-20 10:05:40.101283-06	0.007276	select relpages.reltuple...	1	edbuser
4	2018-08-20 10:06:08.049678-06	9.859956	SELECT name FROM (...)	1	edbuser
5	2018-08-20 10:06:20.599246-06	0.973927	SELECT name FROM (...)	1	edbuser
6	2018-08-20 10:07:32.925681-06	0.001165	select 506*1+20000*0....	1	edbuser
7	2018-08-20 10:08:09.877316-06	0.422506	select * from emp;	14	edbuser
8	2018-08-20 10:08:15.889065-06	0.258003	select * from dept;	4	edbuser
9	2018-08-20 10:08:32.412377-06	10.818188	select * from customers w...	1	edbuser

At the bottom, there is a SQL editor with tabs for 'SQL Query' (selected), 'Metrics', 'Graphical Plan', and 'Text-Based Plan'. The SQL query entered is:

```
select * from customers where customerid=10
```

SQL TUNING.

Reviewing Explain Plans.

- Examinar los diversos costos en diferentes niveles en el plan.
- Buscar escaneos secuenciales en tablas grandes.
 - Comprobar si un tipo de unión es apropiado para el número de filas devueltas.
 - Comprobar los índices utilizados en el plan.
- Revisar si las vistas se utilizan de manera eficiente.

SQL TUNING.

Optimizer Statistics.

- The EDB Postgres Advanced Server Optimizer and Planner usa estadísticas para generar los query plan.
- Elige query plans para ver las tablas de estadísticas.
- Tablas de Estadísticas.
 - Almacenada en tablas de catalogo como: pg_class, pg_stats etc.
 - Almacena información de samples

SQL TUNING.

Updating Planner Statistics.

Tabla de Estadísticas

- No es actualizable en tiempo real, hay que hacerlo de forma manual despues de una carga fuerte de operaciones.
- Puede ser actualizado usando el comando “ANALYZE” o el commando “vacuumdb” con la opción “-z”.
- Almacenado en “pg_class” y “pg_statistics”.
- El commando “ANALYZE” lo puedes correr desde edb-psql sobre tablas específicas.
- El Autovacuum corre ANALYZE configurado por default:

ANALYZE:

- =# ANALYZE [VERBOSE] [table name [(column name [, ...])

SQL TUNING.

Controlling Statistics Collection.

- EDB Postgres recopila y mantiene a nivel de tabla y columna las estadísticas.
- Multi-Column Optimizador.
 - Usando CREATE STATISTICS, ALTER STATISTICS y DROP STATISTICS.
- El nivel de control de estadísticas se puede controlar:

```
=# ALTER TABLE <table> ALTER COLUMN <column>
```
- SET STATISTICS <number>;
- El número puede ser entre 1 y 10000

SQL TUNING.

TABLE SAMPLE Clause.

- Los Samples o registros son extraídos de forma aleatoria desde una table.
- Se utiliza mucho con implementaciones Big Data.

Sintaxis:

- TABLESAMPLE sampling method (argument [, ...]1) [REPEATABLE (seed)]
- Soporta métodos SYSTEM Y BERNOULLI.
 - SYSTEM usa random I/O a nivel página.
 - BERNOULLI usa secuencialmente I/O, scans full hacia las tablas de forma aleatoria.

SQL TUNING.

EJEMPLO TABLE SAMPLE Clause.

- Crea una tabla e inserta datos

```
edb=# CREATE TABLE ts_test (id SERIAL PRIMARY KEY,
title TEXT);
CREATE TABLE
edb=# INSERT INTO ts_test (title)
SELECT 'Record #' || i FROM generate_series(1,1000000) i;
INSERT 0 1000000
```

- Crea una extensión usando “tsm_System_rows”:

```
edb=# CREATE EXTENSION tsm_system_rows;
CREATE EXTENSION
edb=#
-----
```

- Analiza el plan :

```
edb=# EXPLAIN ANALYZE SELECT * FROM ts_test TABLESAMPLE SYSTEM_ROWS(10);
QUERY PLAN
-----
Sample Scan on ts_test  (cost=0.00..4.10 rows=10 width=18) (actual time
=0.048..0.049 rows=10 loops=1)
  Sampling: system_rows ('10'::bigint)
  Planning time: 0.183 ms
  Execution time: 0.066 ms
(4 rows)

edb=#
-----
```

SQL TUNING.

Controlling the Optimizer.

- Parámetro de optimización OPTIMIZER_MODE.
- Usa ALTER SESSION para cambiar el OPTIMIZER_MODE.

Hint	Descripción
ALL_ROWS	Todas las filas.
CHOOSE	Default.
FIRST_ROWS	Optimiza las primeras filas.
FIRST_ROWS_10	Optimiza las primeras 10 filas.
FIRST_ROWS_100	Optimiza las primeras 100 filas.
FIRST_ROWS_1000	Optimiza las primeras 1000 filas.
FIRST_ROWS (n)	Optimiza las primeras n filas de un resultado.

SQL TUNING.

Optimizer Hints.

- Optimiza todos los hints que son directivas embedded seguidas de un DELETE, INSERT, SELECT or UPDATE.
- Optimiza los hints que pueden alterar un plan de ejecución.
- Optimiza los hints que son usados para una determinada selección del plan de ejecución.
- Optimiza hints que influyen en decisiones de velocidad.

SQL TUNING.

Embedding Optimizer Hints.

- El optimizador de hints puede ser usado en dos formas.

```
{ DELETE | INSERT | SELECT | UPDATE } /*+ { hint [ comment ] } [...] */ statement_body
```

```
{ DELETE | INSERT | SELECT | UPDATE } --+ { hint [ comment ] } [...] statement_body
```

SQL TUNING.

Access Method Hints.

- Los siguientes Hints influyen en la forma de optimizar un resultado.

Hint	Description
FULL (table)	Perform a full sequential scan on table
INDEX (table [index] [...])	Use index on table to access the relation
NO_INDEX (table [index] [...])	Do not use index on table to access the relation

El FULL hint es usado para forzar un escaneo secuencial en lugar de usar uno indexado.

- EXPLAIN SELECT /*+ FULL (accounts) */ * FROM accounts WHERE aid = 100;***

Este parámetro NO_INDEX hint también forza un escaneo secuencial.

SQL TUNING.

Join Hints.

- Hay tres posibles planes que pueden ser usados para ejecutar un “join” entre dos tablas.
 - Nested Loop Join.
 - Merge Sort Join.
 - Hash Join.

Hint	Description
USE_HASH(table [...])	Use a hash join with a hash table created from the join attributes of table
NO_USE_HASH(table [...])	Do not use a hash join created from the join attributes of table
USE_MERGE(table [...])	Use a merge sort join for table
NO_USE_MERGE(table [...])	Do not use a merge sort join for table
USE_NL(table [...])	Use a nested loop join sort for table
NO_USE_NL(table [...])	Do not use a nested loop join sort for table

Ejemplo:

```
=# EXPLAIN SELECT /*+ USE_HASH(a) */ b.bid, a.aid, abalance
   FROM branches b, accounts a WHERE b.bid = a.bid;
```

SQL TUNING.

Append Optimizer Hint.

- Por Default Advanced Server agregará nuevos datos en el primer espacio disponible.
- Puedes usar APPEND para optimizar un hint con INSERT o un SELECT.
- Es útil con una carga de datos grande.

Ejemplo:

```
=# INSERT /*+APPEND*/ INTO sales VALUES (10, 10, '01I-SEP-2014', 10,  
'OR');  
  
=# INSERT INTO sales history SELECT /*+APPEND*/ FROM sales;
```

SQL TUNING.

Parallelism Hints.

- Parallel Scanning ofrece una mejora en el performance en un escaneo secuencial.
- El PARALLEL optimiza un hint y forca el escaneo.
- EL NO_PARALLEL optimiza un hint y previene de un uso en paralelo en un escaner.
- PARALLEL (table [parallel degree | DEFAULT])
• ~~Sintaxis:~~
NO_PARALLEL (table)

Ejemplo:

```
=# EXPLAIN SELECT /*+ PARALLEL (pgbench_accounts) */  
* FROM pgbench_accounts;
```

SQL TUNING.

Restructuring SQL Statements.

- Reescribir SQL ineficiente suele ser más fácil que repararlo y evite la conversión de tipos implícita
- Evite las expresiones ya que el optimizador puede ignorar los índices en tablas y columnas
- Evite los escaneos completos de tablas
- Use equijoins siempre que sea posible para mejorar la eficiencia de SQL
- Use exploraciones de consultas paralelas para mejorar el rendimiento de las consultas

SQL TUNING.

General Indexing Guidelines.

- Crea índices cada vez que los necesites.
- Remueve índices que no uses.
- Agrega un índice por cada query SQL lenta.
- Verifica el uso de los índices con el comando EXPLAIN.

SQL TUNING.

Indexes.

EDB Postgres Advanced Server ofrece diferentes tipos de Index:

- **B-tree** -El rango de query puede ser sorteada (Default index type).
- **Hash** - Comparación.
- **GIST** - Puede ser usado dependiendo de la estrategia de indexación.
- **SP-GiST** - Soporta diferentes tipos de búsqueda.
- **GIN** - Se ocupa en arreglos.
- **BRIN (Block Range Index)** - Acelera el escaneo en tablas grandes.

SQL TUNING.

Unique Indexes.

- Un índice puede ser definido en una o más columnas de una tabla.
- Actualmente, solo B-tree , GIST and GIN soportan multicolumna.
- **Ejemplo:**

```
=# CREATE INDEX test_idx1 ON test (id_1, id_2);
```

Este índice será usado:

```
=# SELECT * FROM test WHERE id_1=1880 AND id_2= 4500;
```

SQL TUNING.

Indexes and ORDER BY.

- Puedes ajustar el orden de B-tree con las opciones ASC, DESC, NULLS FIRST, y NULLS.
- LAST Al crear un indice ahoora tiempo en la organización.
- Por default, B-tree indexa de forma ascendente.
- Examples:

```
=# CREATE INDEX test2_info_nulls_low ON test2 (info NULLS FIRST);
```

```
=# CREATE INDEX test3_desc_index ON test3 (id DESC NULLS LAST);
```

SQL TUNING.

Functional Indexes.

- Un índice puede ser creado con un valor de una columna.
- Ejemplo:

```
=# CREATE INDEX test1_lower_col1_idx ON test1
(lower(col1));
```

- Las expresión Index es relativamente costoso y difícil de mantener.
- Los índices en expresiones son de mucha ayuda cuando la velocidad es lo importante.
- Los índices pueden ser creados en funciones.

SQL TUNING.

BRIN Indexes Indexes.

- Disponible en EDB Postgres Advanced Server desde la versión 9.5.
- BRIN (Índice de rango de bloque) almacena metadatos en el rango (mínimo y máx.) de páginas.
- El rango de bloque es por defecto de 1 MB y pages_per_range.
- El parámetro de almacenamiento determina el tamaño del rango de bloques.
- Funciones brin_summarize_range() y brin_desummarize_range () actualiza y elimina.
- Resumen del índice BRIN para un rango específico.
- BRIN es de tamaño pequeño, más fácil de mantener y elimina el disco.
- I/O almacenando el resumen de la distribución de datos en el disco.
- Diseñado para manejar grandes columnas de tabla, que se correlacionan con el ubicación física dentro de la tabla.

SQL TUNING.

BRIN Indexes Ejemplo:

```
edb=# CREATE TABLE brin_example AS
edb-# SELECT generate_series(1,100000000) AS id;
edb=#
edb=# CREATE INDEX brin_index ON brin_example USING brin(id);
edb=#
edb=# CREATE INDEX btree_index ON brin_example(id);
edb=#
edb=# SELECT relname, pg_size.pretty(pg_relation_size(oid))
edb-# FROM pg_class
edb-# WHERE relname LIKE 'brin_%' OR relname='btree_index'
edb-# ORDER BY relname;
   relname    | pg_size.pretty
-----+-----
brin_example | 3457 MB
brin_index   | 104 kB
btree_index  | 2142 MB
(3 rows)
```

SQL TUNING.

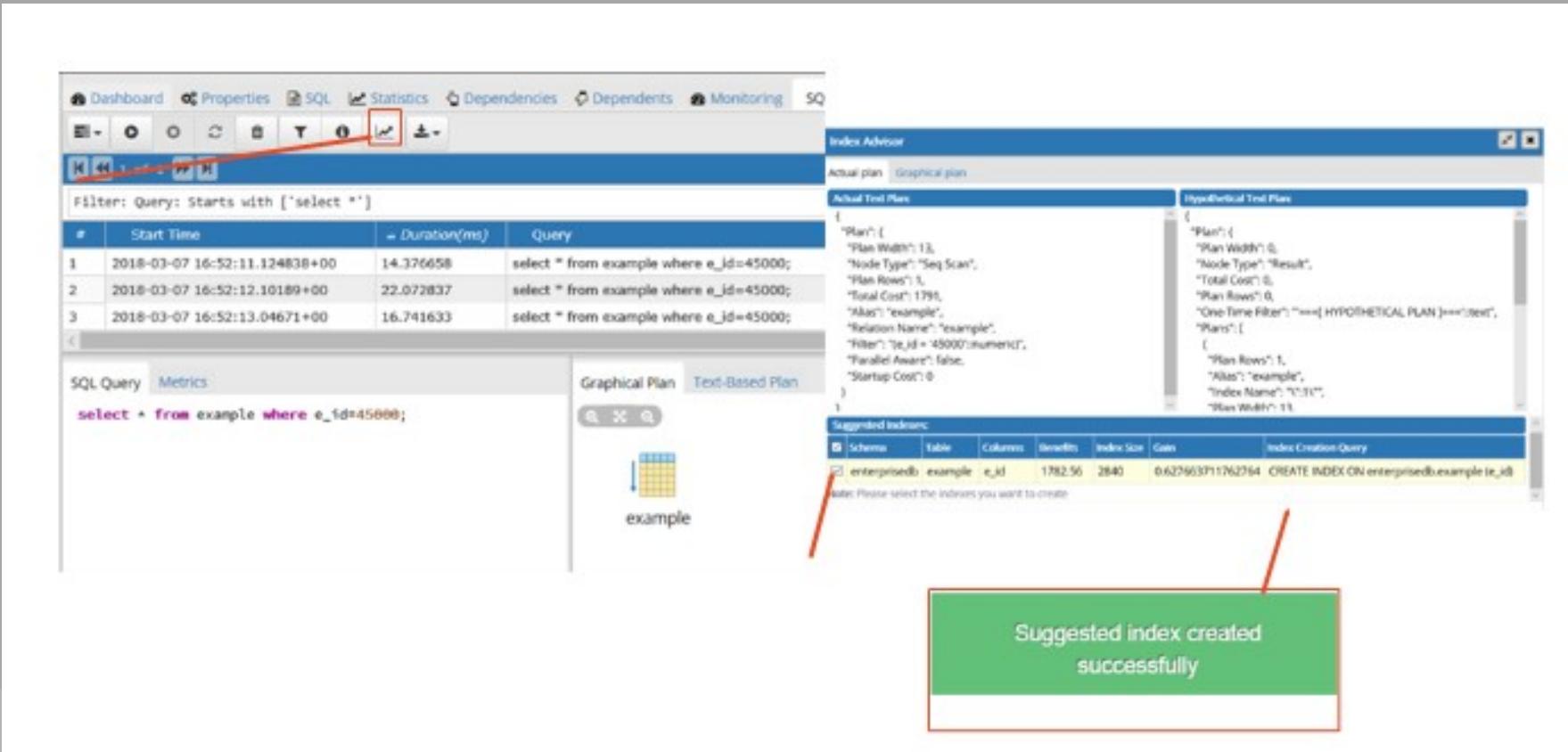
Examining Indexes Usage

Es difícil formular un procedimiento general para determinar qué índices crear.

- Siempre ejecute ANALYZE primero.
- Usar datos reales para la experimentación.
- Cuando no se usan índices, puede ser útil para probar y forzar su uso.
- El comando EXPLAIN ANALYZE puede ser útil en estos casos.

SQL TUNING.

PEM Index Advisor



The screenshot illustrates the PEM Index Advisor interface. On the left, a timeline shows three recent queries, all of which are 'select * from example where e_id=45000'. The timeline bar is highlighted with a red box. Below the timeline, a SQL query is entered:

```
select * from example where e_id=45000;
```

The graphical plan for this query is shown, indicating a 'Seq Scan' operation. A red arrow points from the timeline bar to the graphical plan area.

In the center, the 'Index Advisor' window is open. It displays the 'Actual Test Plan' and 'Hypothetical Test Plan' side-by-side. The 'Actual Test Plan' shows a 'Seq Scan' with a cost of 1791. The 'Hypothetical Test Plan' shows a 'Result' node with a cost of 0, indicating an index could potentially eliminate the scan. A red arrow points from the 'Actual Test Plan' to the 'Hypothetical Test Plan'.

On the right, a 'Suggested Indexes' table lists an index for the 'example' table on the 'e_id' column. The table includes columns for Scheme, Table, Column, Benefits, Index Size, Gain, and Index Creation Query. The entry is:

Scheme	Table	Column	Benefits	Index Size	Gain	Index Creation Query
enterprisedb	example	e_id	1782.56	2840	0.622663711262764	CREATE INDEX ON enterprisedb.example(e_id)

A red arrow points from the 'Index Creation Query' in the table to a green box at the bottom right containing the message:

Suggested index created successfully

SQL TUNING.

Ejemplos:

```
edbstore=> \d test
              Table "edbuser.test"
 Column |      Type       | Collation | Nullable | Default
-----+-----+-----+-----+-----+
 id    | numeric
 name  | character varying
 city  | character varying

edbstore=> explain select * from test where id=3500000;
               QUERY PLAN
-----
-----
Gather  (cost=1000.00..58889.72 rows=1 width=18)
  Workers Planned: 2
    -> Parallel Seq Scan on test  (cost=0.00..57889.62 rows=1 width=18)
        Filter: (id = '3500000'::numeric)
Result  (cost=0.00..0.00 rows=0 width=0)
  One-Time Filter: '===[ HYPOTHETICAL PLAN ]==='::text
    -> Index Scan using "<hypothetical-index>:1" on test  (cost=0.56..8.57
rows=1 width=18)
        Index Cond: (id = '3500000'::numeric)
(8 rows)

edbstore=> █
```

SQL TUNING.

Ejemplos:

```
edbstore=> explain analyze select * from test where id=3500000;
               QUERY PLAN
-----
Gather  (cost=1000.00..58889.72 rows=1 width=18) (actual time=534.267..538
.153 rows=1 loops=1)
  Workers Planned: 2
  Workers Launched: 2
    -> Parallel Seq Scan on test  (cost=0.00..57889.62 rows=1 width=18) (ac
tual time=439.770..519.251 rows=0 loops=3)
        Filter: (id = '3500000'::numeric)
        Rows Removed by Filter: 1666666
  Execution time: 538.483 ms
  Result  (cost=0.00..0.00 rows=0 width=0)
    One-Time Filter: '===[ HYPOTHETICAL PLAN ]==='::text
      -> Index Scan using "<hypothetical-index>:4" on test  (cost=0.56..8.57
rows=1 width=18)
          Index Cond: (id = '3500000'::numeric)
(11 rows)

edbstore=>
```

```
edbstore=> create index idx_test_id on test(id);■
```

SQL TUNING.

Ejemplos:

```
edbstore=> create index idx_test_id on test(id);
CREATE INDEX
edbstore=> analyze test;
ANALYZE
edbstore=> explain analyze select * from test where id=3500000;
               QUERY PLAN

-----
-----
      Index Scan using idx_test_id on test  (cost=0.43..8.45 rows=1 width=18) (actual
      time=0.979..0.980 rows=1 loops=1)
        Index Cond: (id = '3500000'::numeric)
      Execution time: 2.613 ms
(3 rows)

edbstore=>
```

SQL TUNING.

Ejemplos:

```
Index Scan using idx test id on test  (cost=0.43..8.45 rows=1 width=18) (a
ctual time=0.979..0.980 rows=1 loops=1)
  Index Cond: (id = '3500000'::numeric)
  Execution time: 2.613 ms
(3 rows)

edbstore=> drop index idx_test_id ;
DROP INDEX
edbstore=> create index idx_test_id on test(id) where id between 3000000 an
d 4000000;
CREATE INDEX
edbstore=> analyze test;
ANALYZE
edbstore=> explain analyze select * from test where id=3500000;
                                         QUERY PLAN

-----
-----
Index Scan using idx_test_id on test  (cost=0.42..8.44 rows=1 width=18) (a
ctual time=8.951..8.952 rows=1 loops=1)
  Index Cond: (id = '3500000'::numeric)
  Execution time: 9.028 ms
(3 rows)

edbstore=>
```

SQL TUNING.

Ejemplos:

```
Index Scan using idx_test_id on test  (cost=0.42..8.44 rows=1 width=18)
actual time=0.019..0.020 rows=1 loops=1
  Index Cond: (id = '3500000'::numeric)
  Execution time: 0.094 ms
(3 rows)

edbstore=> explain analyze select * from test where id=3500000;
               QUERY PLAN

-----
-----
Index Scan using idx_test_id on test  (cost=0.42..8.44 rows=1 width=18)
actual time=0.014..0.014 rows=1 loops=1
  Index Cond: (id = '3500000'::numeric)
  Execution time: 0.066 ms
(3 rows)

edbstore=> \di+ idx*
              List of relations
 Schema |      Name      | Type | Owner | Table | Size | Description
-----+-----+-----+-----+-----+-----+
edbuser | idx_test_id | index | edbuser | test  | 21 MB |
(1 row)

edbstore=> █
```

SQL TUNING.

Last Step – Review the Final Plan

- Vuelve a comprobar si faltan índices.
- Verifique que las estadísticas de la tabla muestren estimaciones correctas.
- Verifique los escaneos secuenciales de tablas grandes.
- Verificar el uso de sugerencias del optimizador.
- Compare el costo del primer y último plan.

- Eres el DBA y deberás lanzar Querys desde PEM.

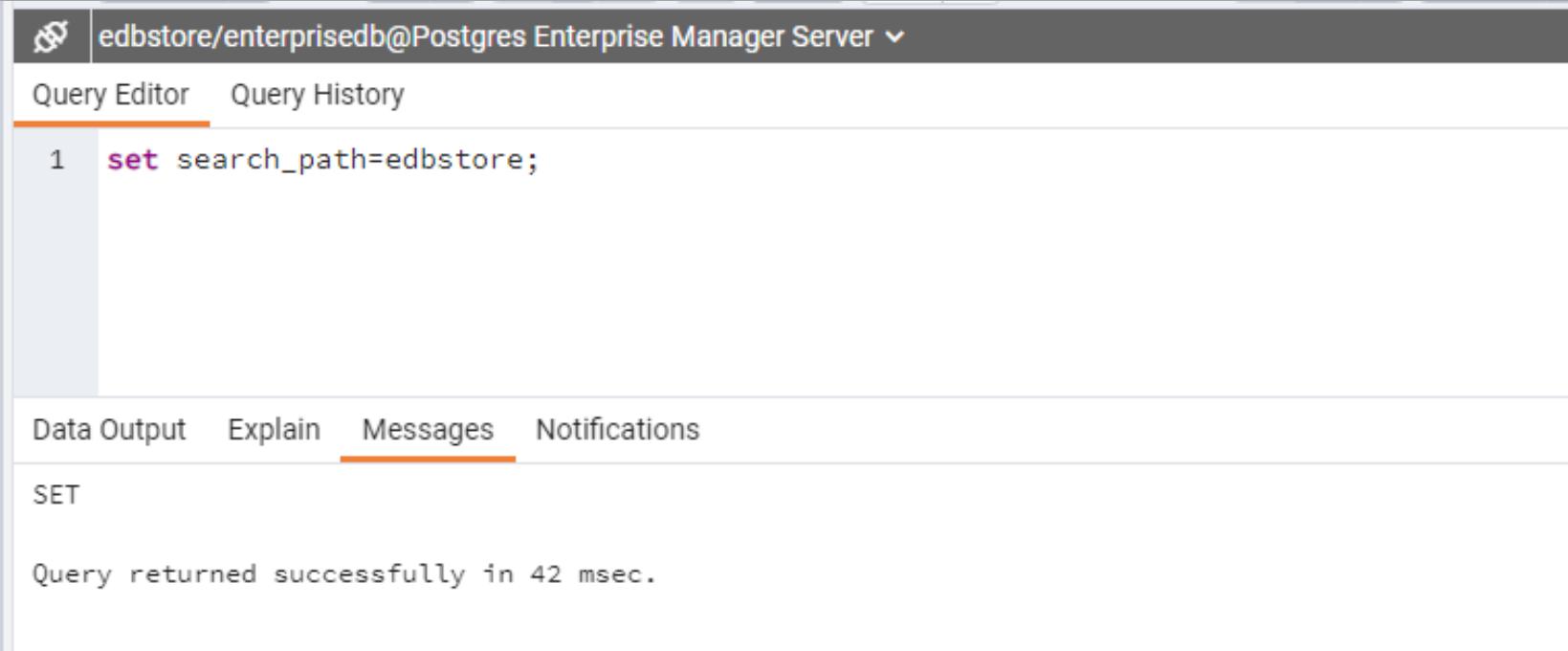
Ejecuta la siguiente Query:

```
SELECT * FROM customers JOIN orders  
USING (customerid) ;
```

LABORATORIO

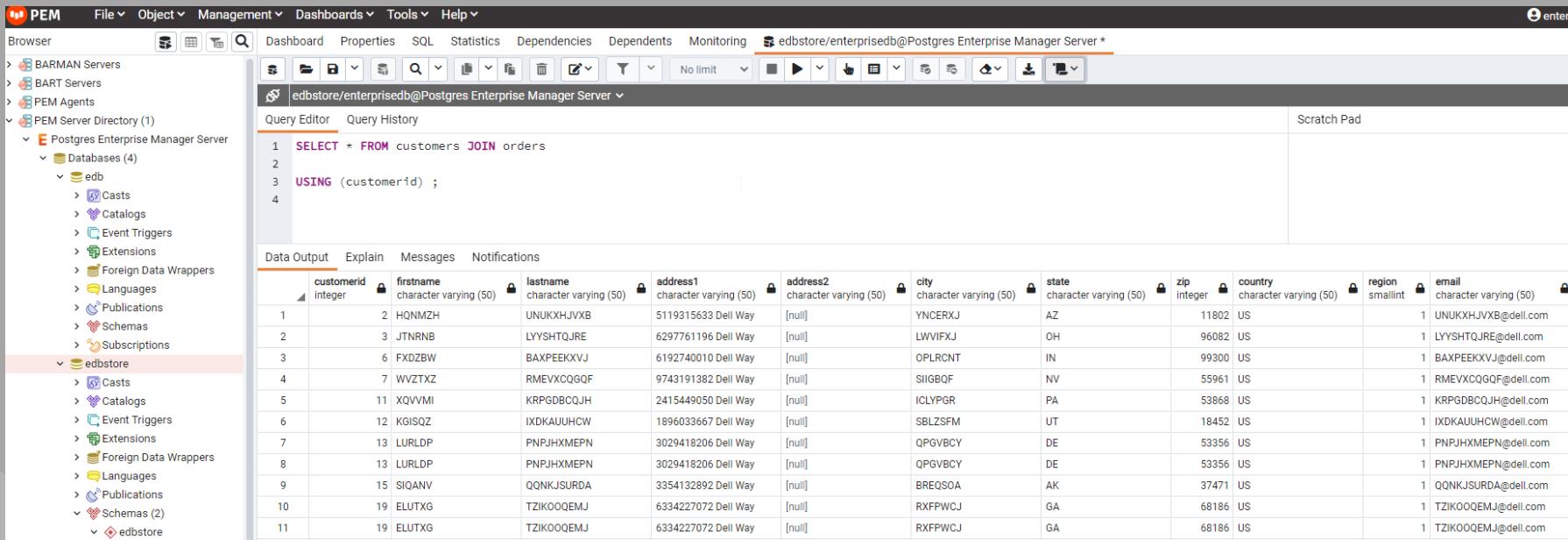
Instalación

- Eres el DBA y deberás lanzar Querys desde PEM.



The screenshot shows the Postgres Enterprise Manager (PEM) interface. The title bar reads "edbstore/enterprisedb@Postgres Enterprise Manager Server". The main area is the "Query Editor" tab, which is active. A single line of SQL code is present: "1 set search_path=edbstore;". Below the editor, there are tabs for "Data Output", "Explain", "Messages", and "Notifications", with "Messages" being the active tab. The message content shows the output of the query: "SET" and "Query returned successfully in 42 msec."

- Eres el DBA y deberás lanzar Querys desde PEM.



The screenshot shows the Postgres Enterprise Manager (PEM) interface. The left sidebar displays the 'Browser' tree, which includes sections for BARMAN Servers, BART Servers, PEM Agents, PEM Server Directory (with Postgres Enterprise Manager Server and edbstore databases selected), and various database objects like Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, and Subscriptions. The main area features a 'Query Editor' tab where the following SQL query is entered:

```

1 SELECT * FROM customers JOIN orders
2
3 USING (customerid) ;
4

```

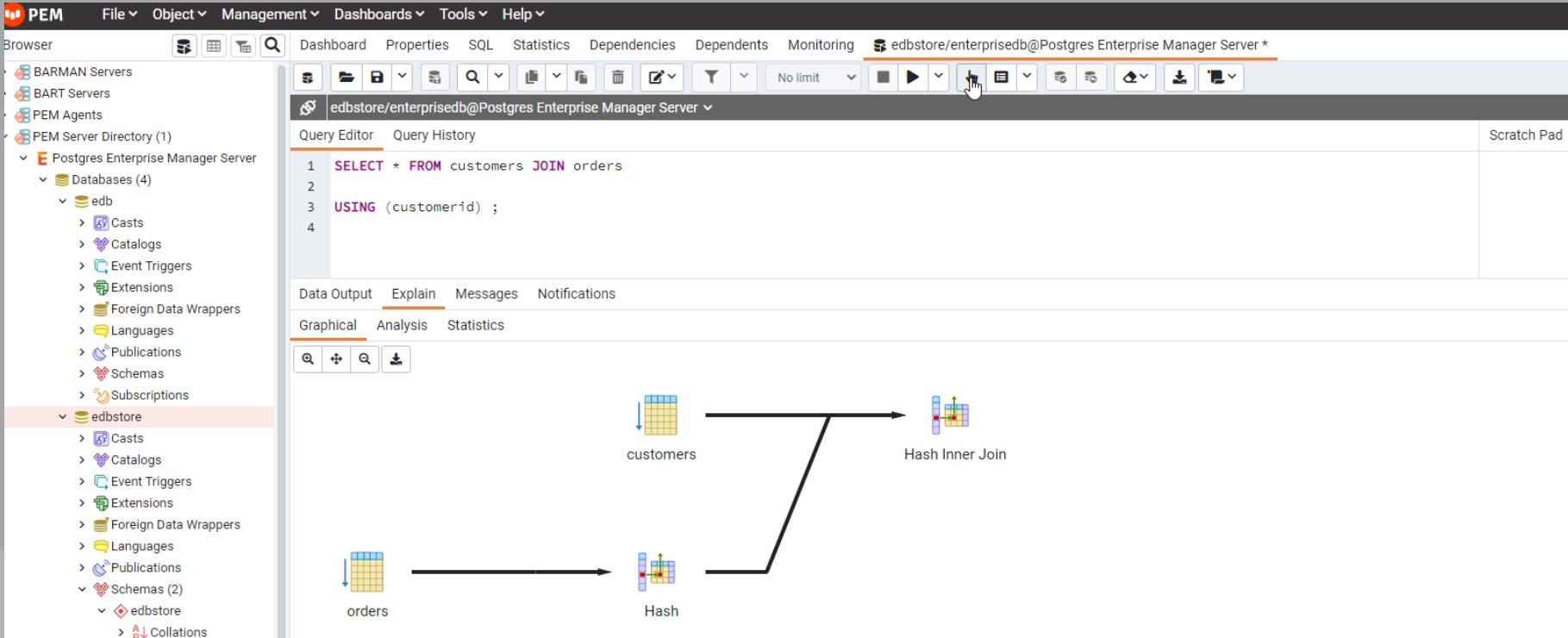
Below the query editor is a 'Data Output' table displaying customer data from the 'customers' and 'orders' tables. The columns are:

	customerid	firstname	lastname	address1	address2	city	state	zip	country	region	email
1	2	HQNMZH	UNUKXHJVXB	5119315633 Dell Way	[null]	YNCERXJ	AZ	11802	US	1	UNUKXHJVXB@dell.com
2	3	JTNRNB	LYSHTQRE	6297761196 Dell Way	[null]	LWVIFXJ	OH	96082	US	1	LYSHTQRE@dell.com
3	6	FXDZBW	BAXPEEKVJ	6192740010 Dell Way	[null]	OPLRCNT	IN	99300	US	1	BAXPEEKVJ@dell.com
4	7	WVZTXZ	RMEVXCQGQF	9743191382 Dell Way	[null]	SIIGBQF	NV	55961	US	1	RMEVXCQGQF@dell.com
5	11	XOVVMI	KRPGDBCQJH	2415449050 Dell Way	[null]	ICLYPGR	PA	53868	US	1	KRPGDBCQJH@dell.com
6	12	KGISQZ	IXDKAUUHCW	1896033667 Dell Way	[null]	SBLZSFM	UT	18452	US	1	IXDKAUUHCW@dell.com
7	13	LURLDP	PNPJHXMEPN	3029418206 Dell Way	[null]	QPGBCY	DE	53356	US	1	PNPJHXMEPN@dell.com
8	13	LURLDP	PNPJHXMEPN	3029418206 Dell Way	[null]	QPGBCY	DE	53356	US	1	PNPJHXMEPN@dell.com
9	15	SIDANV	QQNKJSURDA	3354132892 Dell Way	[null]	BREGSOA	AK	37471	US	1	QQNKJSURDA@dell.com
10	19	ELUTXG	TZIKOOQEMJ	6334227072 Dell Way	[null]	RXFPWCJ	GA	68186	US	1	TZIKOOQEMJ@dell.com
11	19	ELUTXG	TZIKOOQEMJ	6334227072 Dell Way	[null]	RXFPWCJ	GA	68186	US	1	TZIKOOQEMJ@dell.com

LABORATORIO

Instalación

- Eres el DBA y deberás lanzar Querys desde PEM.



The screenshot shows the Postgres Enterprise Manager (PEM) interface. The left sidebar displays a tree view of database objects under 'PEM Server Directory (1)'. The 'edbstore' database is selected. The main area contains a 'Query Editor' tab with the following SQL code:

```
1 SELECT * FROM customers JOIN orders
2
3 USING (customerid) ;
4
```

Below the query editor, there is an 'Explain' tab which displays a graphical execution plan. The plan shows two tables, 'customers' and 'orders', being hashed and then joined via a 'Hash Inner Join'.

```
graph LR
    subgraph Explain_Plan [Execution Plan]
        direction TB
        C[customers] --> H1[Hash]
        O[orders] --> H1
        H1 --> HJ[Hash Inner Join]
        HJ --> J[Join]
    end
```

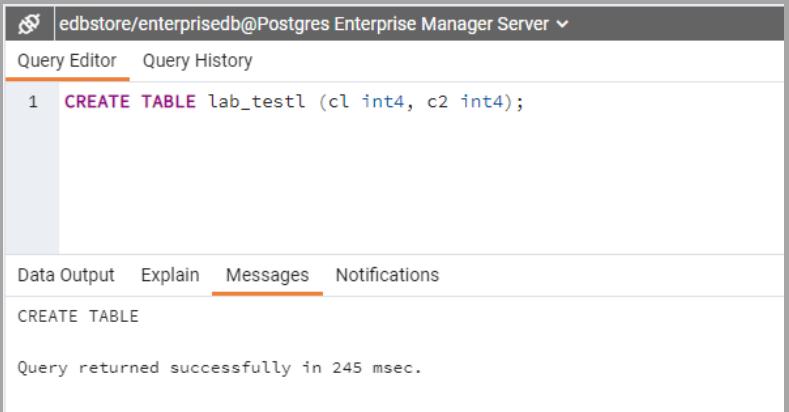
- CREA LA SIGUIENTE TABLA.

- =# CREATE TABLE lab_test1 (cl int4, c2 int4);
- =# INSERT INTO lab_test1(cl, c2)
VALUES (generate_series (1, 100000), 1);
- =# INSERT INTO lab_test1(cl, c2)
VALUES (generate_series (100001, 200000), 2);
- =# INSERT INTO lab_test1(cl, c2)
VALUES (generate_series (200001, 300000), 3);

LABORATORIO

Instalación

- CREA LA SIGUIENTE TABLA.



edbstore/enterprisedb@Postgres Enterprise Manager Server

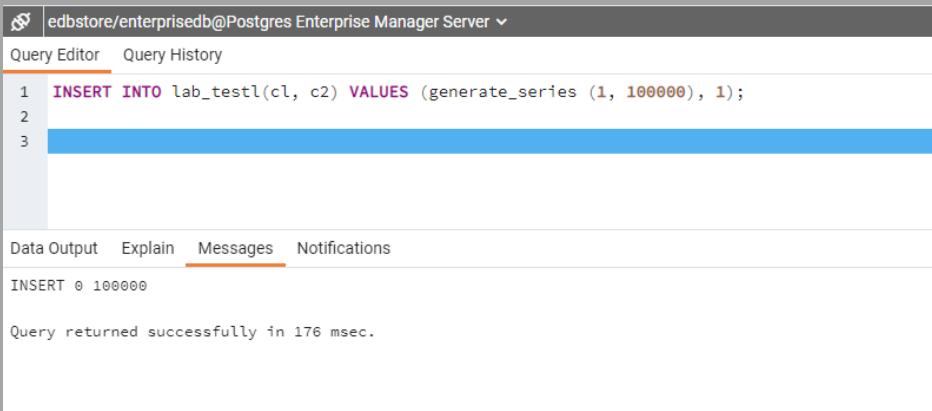
Query Editor Query History

```
1 CREATE TABLE lab_testl (cl int4, c2 int4);
```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 245 msec.



edbstore/enterprisedb@Postgres Enterprise Manager Server

Query Editor Query History

```
1 INSERT INTO lab_testl(cl, c2) VALUES (generate_series (1, 100000), 1);  
2  
3
```

Data Output Explain Messages Notifications

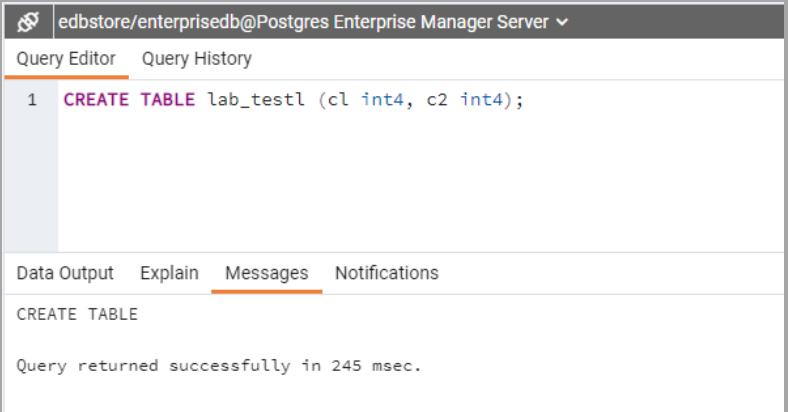
INSERT 0 100000

Query returned successfully in 176 msec.

LABORATORIO

Instalación

- CREA LA SIGUIENTE TABLA.



edbstore/enterprisedb@Postgres Enterprise Manager Server

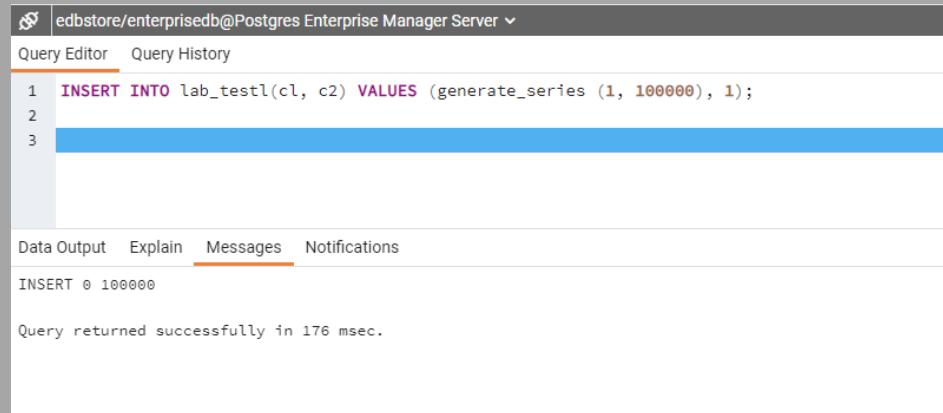
Query Editor Query History

```
1 CREATE TABLE lab_testl (cl int4, c2 int4);
```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 245 msec.



edbstore/enterprisedb@Postgres Enterprise Manager Server

Query Editor Query History

```
1 INSERT INTO lab_testl(cl, c2) VALUES (generate_series (1, 100000), 1);
2
3
```

Data Output Explain Messages Notifications

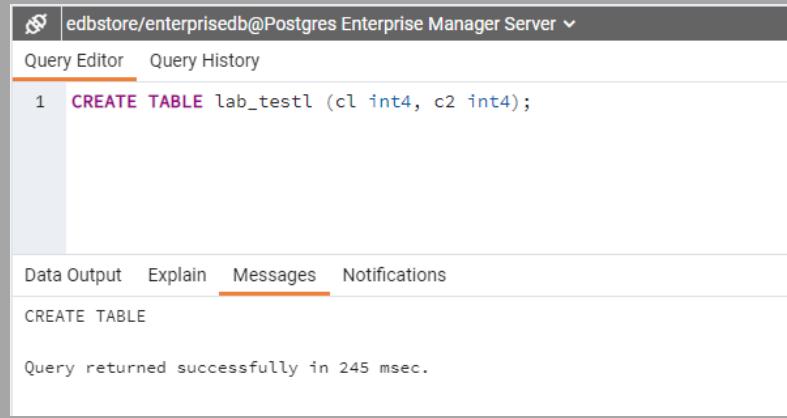
INSERT 0 100000

Query returned successfully in 176 msec.

LABORATORIO

Instalación

- CREA LA SIGUIENTE TABLA.



edbstore/enterprisedb@Postgres Enterprise Manager Server

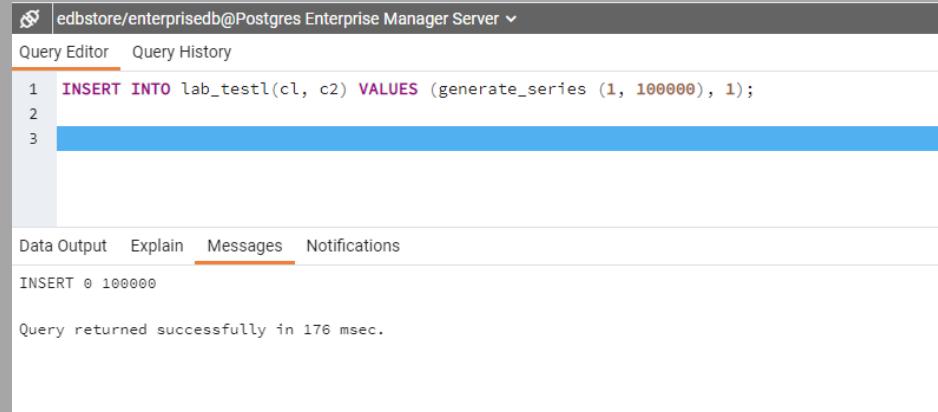
Query Editor Query History

```
1 CREATE TABLE lab_testl (cl int4, c2 int4);
```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 245 msec.



edbstore/enterprisedb@Postgres Enterprise Manager Server

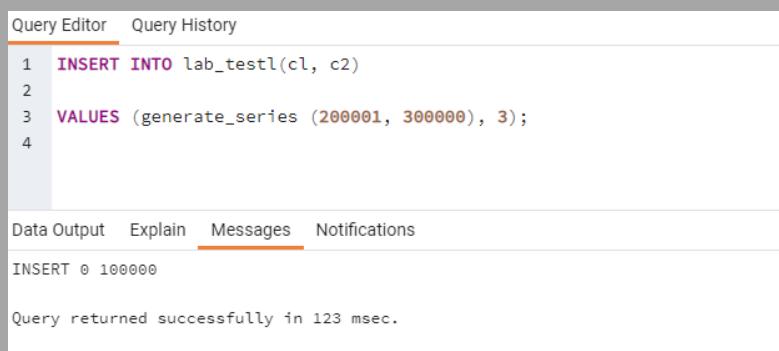
Query Editor Query History

```
1 INSERT INTO lab_testl(cl, c2) VALUES (generate_series (1, 100000), 1);
```

Data Output Explain Messages Notifications

INSERT 0 100000

Query returned successfully in 176 msec.



Query Editor Query History

```
1 INSERT INTO lab_testl(cl, c2)
2
3 VALUES (generate_series (200001, 300000), 3);
```

Data Output Explain Messages Notifications

INSERT 0 100000

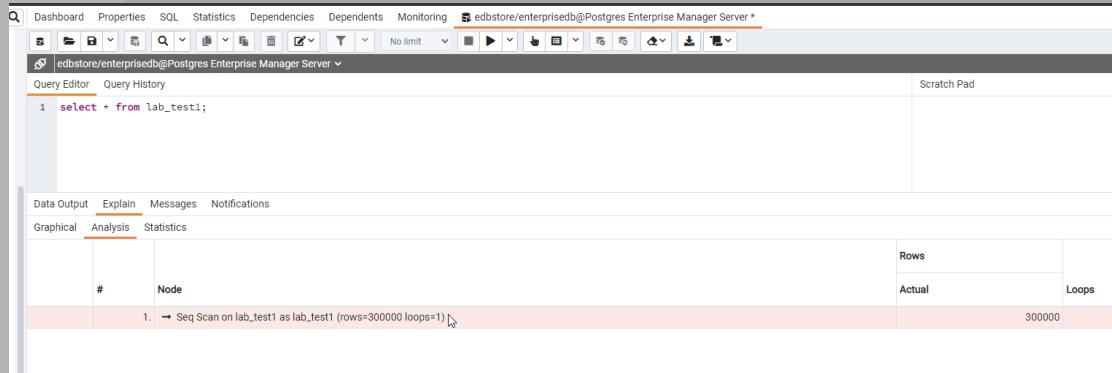
Query returned successfully in 123 msec.

LABORATORIO

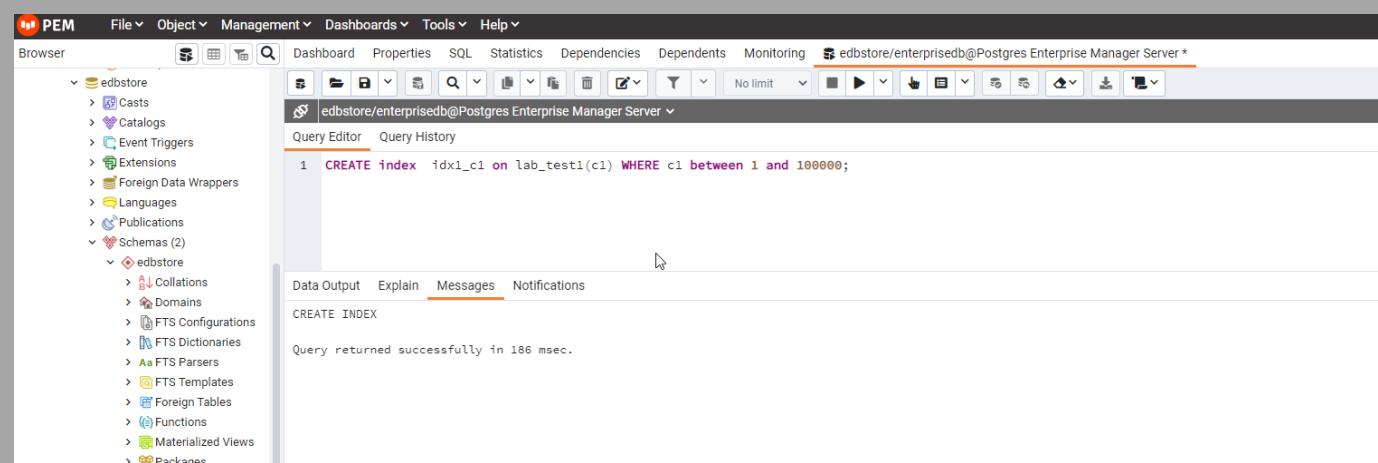
Instalación

- CREA TRES indexes sobre la tabla lab_test1
- Observa el uso

Index Name	Predicate
idx1_c1	c1 between 1 and 100000
idx2_c1	c1 between 100001 and 200000
idx3_c3	c1 between 200001 and 300000



The screenshot shows the Postgres Enterprise Manager Server interface. In the top navigation bar, 'Explain' is selected under the 'Analysis' tab. The main area displays a query plan for a sequential scan on the 'lab_test1' table. The plan shows one node with 300,000 rows scanned. Below the plan, there is a table with columns '#', 'Node', 'Actual', and 'Loops'. The first row has a pink background and shows the node number 1, the node name 'Seq Scan on lab_test1 as lab_test1 (rows=300000 loops=1)', and values for Actual and Loops.



The screenshot shows the Postgres Enterprise Manager Server interface. In the top navigation bar, 'Tools' is selected. The left sidebar shows the 'Browser' tree, which includes the 'edbstore' database node. The main area shows the 'Query Editor' tab with the following SQL command:

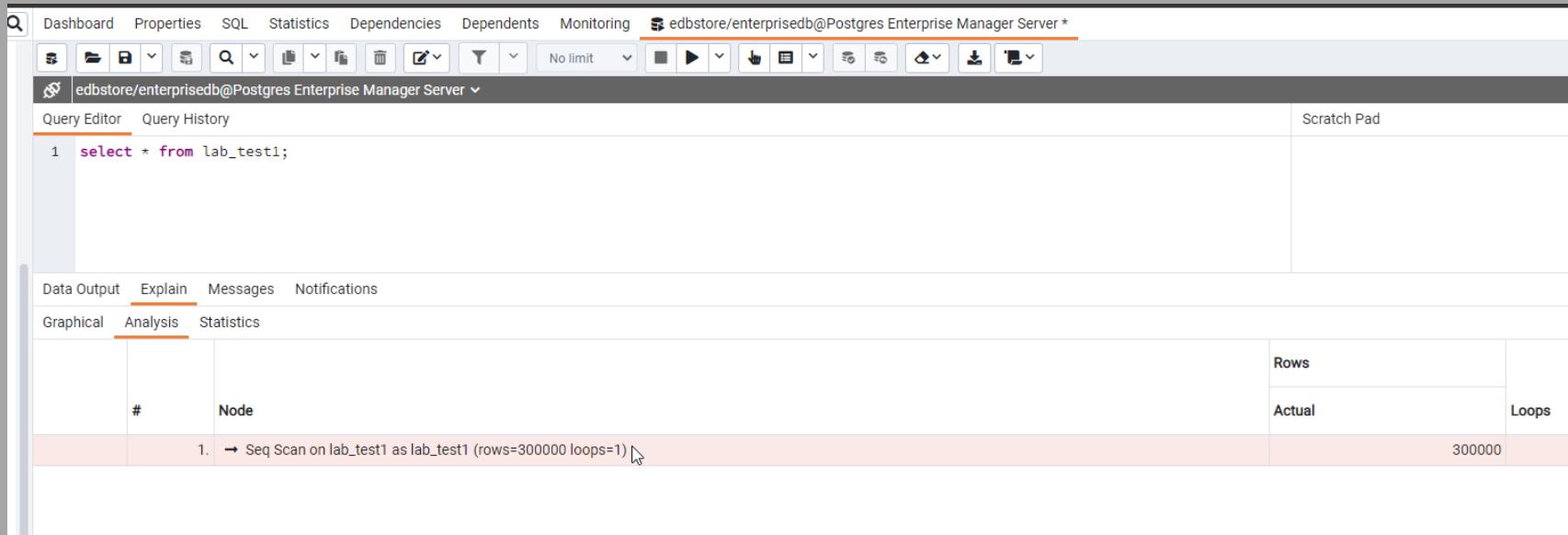
```
1 CREATE index idx1_c1 on lab_test1(c1) WHERE c1 between 1 and 100000;
```

Below the query editor, the 'Data Output' tab is selected, showing the message: "CREATE INDEX". Further down, it says "Query returned successfully in 186 msec."

LABORATORIO

Instalación

- CREA TRES indexes sobre la tabla lab_test1
- Observa el uso



The screenshot shows the pgAdmin III interface with a query editor window. The query is:

```
1 select * from lab_test1;
```

The Explain tab is selected, showing the following execution plan:

#	Node	Rows	Loops
1.	→ Seq Scan on lab_test1 as lab_test1 (rows=300000 loops=1)	300000	

PERFORMANCE TUNING.

Performance Tuning Overview.

- Performance Tuning es un grupo de actividades utilizadas para optimizar una base de datos.
- El ajuste de la base de datos se utiliza para corregir:
 - SQL mal escrito.
 - Mala gestión de sesiones.
 - Parámetros de base de datos mal configurados.
 - Problemas de E/S del sistema operativo.
 - Sin mantenimiento de base de datos.

PERFORMANCE TUNING.

Performance Tuning Overview.

Recolección de información.

- Servidores.
 - * CPU: modelo, velocidad, número de CPUs, arquitectura.
 - * RAM: cantidad, velocidad.
 - *Servidor dedicado a PostgreSQL o compartido.
- Almacenamiento
 - *Tipos de interfaces (controladores, RAID).
 - *Discos: tamaño, velocidad.
- Red
 - *Tipo de red y capacidad de la red.
 - *Tarjetas de red usadas y modelos.
 - *Configuración de switch/router.
 - * ¿Red dedicada entre la aplicación y la BBDD?

PERFORMANCE TUNING.

Performance Tuning Overview.

Recolección de información.

Recolección de información

- Sistema Operativo
 - * Tipo de SO, versión, parches aplicados.
 - * Información sobre controladoras de HW usadas.
 - * ¿Estamos usando la última versión del kernel?.
 - * ¿Existen otras aplicaciones consumiendo recursos del servidor?
- Sistema de ficheros
 - * Tipo de sistema de ficheros utilizado.
 - * Localización de los ficheros del SO, PostgreSQL, y otras aplicaciones.
 - * Configuración del sistema de ficheros utilizado.
 - * ¿Está xlog (pg_wal) en un disco dedicado?

PERFORMANCE TUNING.

Tuning Technique.

**Analisa el Sistema
Operativo**

**Checa la
aplicación**

**Revisa y Tunea
tu BD**

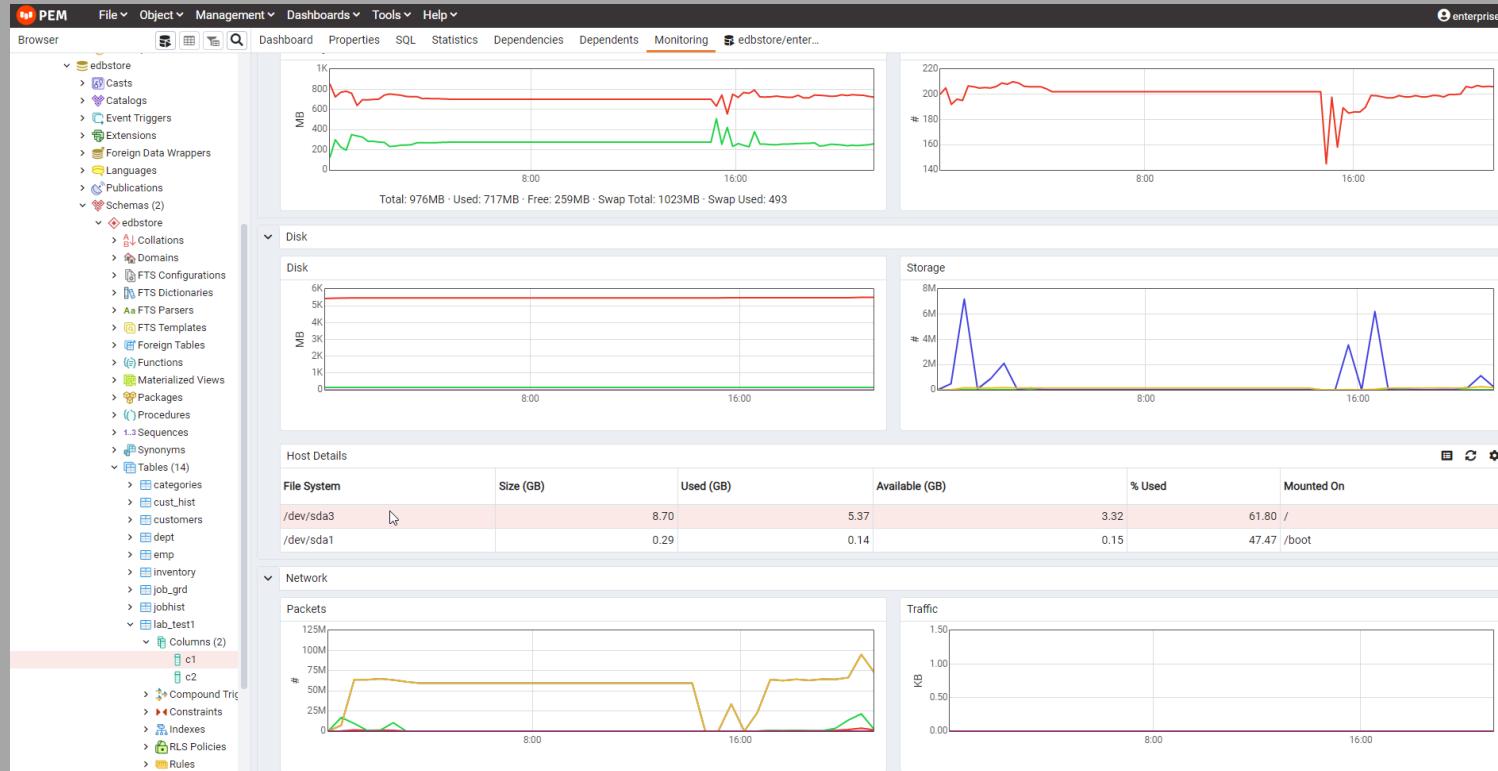
PERFORMANCE TUNING.

PEM CPU GRAPHS.



PERFORMANCE TUNING.

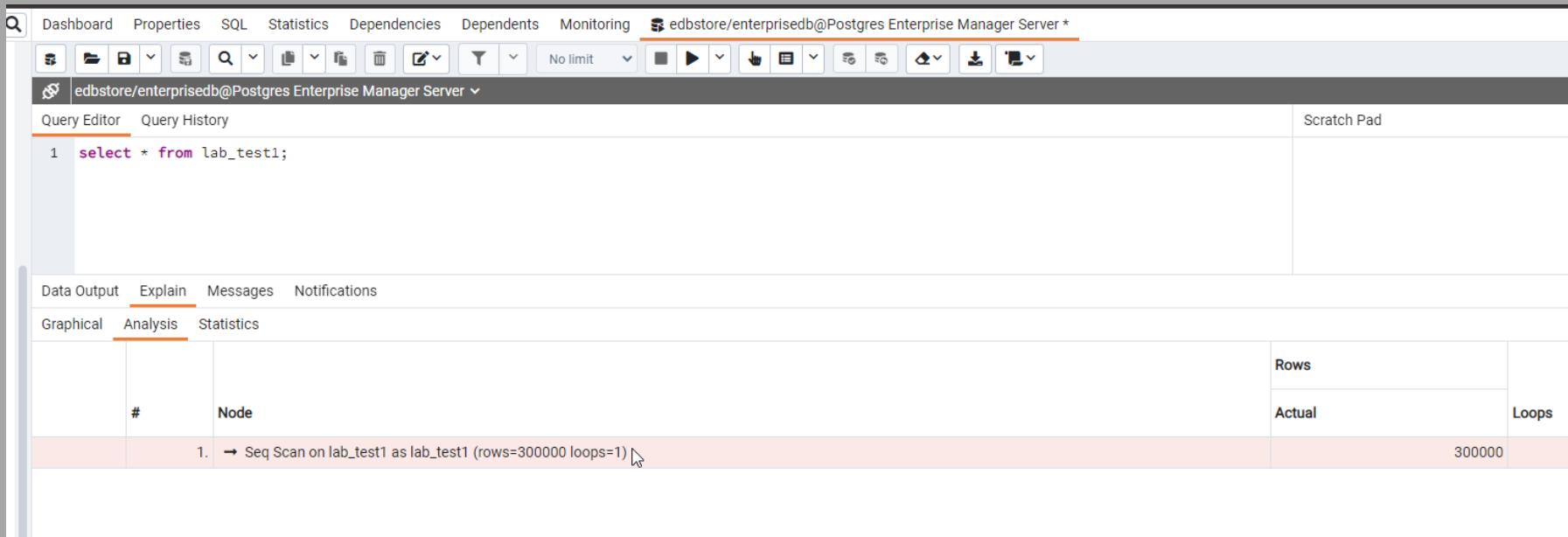
PEM CPU GRAPHS.



LABORATORIO

Instalación

- CREA TRES indexes sobre la tabla lab_test1
- Observa el uso



The screenshot shows a PostgreSQL query execution plan. The query being run is:

```
1 select * from lab_test1;
```

The Explain tab is selected, showing the following plan:

#	Node	Rows	Actual	Loops
1.	Seq Scan on lab_test1 as lab_test1 (rows=300000 loops=1)	300000	300000	1

PERFORMANCE TUNING.

Server Parameter Tuning.

- El archivo más importante es el postgresql.conf.
- Los parámetros de este archivo deben de ser correctamente seteados, de acuerdo a las características del servidor.
- Algunos parámetros requieren un reinicio.
- Información de:
 - Tamaño de la base de datos.
 - Tamaño de tablas.
 - Tipo y frecuencia de querys.
 - Memoria RAM disponible.
 - Número de conexiones concurrentes.

PERFORMANCE TUNING.

PEM – Postgres Expert.

- Postgres Expert analiza el servidor configuración e informes de rendimiento potencial y problemas de seguridad.
- El informe también proporciona sugerencias y mejores prácticas para abordar posibles problemas.
- Postgres Expert es una utilidad de asesoramiento que proporciona consejos sobre rendimiento, seguridad y configuración.

PERFORMANCE TUNING.

PEM – Postgres Expert Report.

Postgres Expert Report

Generated: 2018-03-07 17:10:50

Jump to: EPAS10_Prod1

Summary:

Settings	Value
Number of servers tested:	1
Number of rules checked:	31
Number of High alerts:	1
Number of Medium alerts:	4
Number of Low alerts:	3

Server: EPAS10_Prod1 (10.138.0.2:5444)

Advisor: Configuration Expert

Rule	Database	Severity
Check checkpoint_completion_target	-	Medium
Check effective_cache_size	-	Medium
Check effective_io_concurrency	-	Low
Check reducing_random_page_cost	-	Low

Advisor: Schema Expert

Rule	Database	Severity
Check data and transaction log on same drive	-	High
Check for missing foreign key indexes	edb	Medium
Check for missing primary keys	edb	Low

Advisor: Security Expert

Rule	Database	Severity
Check SSL for improved connection security	-	Medium

Report generated by: Postgres Enterprise Manager™

PERFORMANCE TUNING.

Connections Settings.

max_conexiones.

- Establece el número máximo de conexiones simultáneas.
- Cada conexión de usuario tiene un backend de usuario asociado proceso en el servidor.
- Los procesos de back-end del usuario finalizan cuando un usuario inicia sesión.
- La agrupación de conexiones puede reducir la sobrecarga en postmaster mediante la reutilización de procesos de back-end de usuario existentes.

PERFORMANCE TUNING.

Connections Settings.

shared buffers

- Establece el número de búferes de memoria compartida utilizados por el servidor de base de datos.
- Cada búfer tiene 8K bytes.

El valor mínimo debe ser 16 y al menos $2 \times \text{max_conexiones}$.

- La configuración predeterminada es administrada por dynatune.
- 6% - 25% de la memoria disponible es una buena pauta general.
- Puede encontrar mejores resultados manteniendo la configuración relativamente bajo y usando más el caché del sistema operativo.

PERFORMANCE TUNING.

Connections Settings.

work_mem

- Cantidad de memoria en KB para ser utilizada internamente y hash de tablas antes de cambiar a archivos de disco temporales.
= El valor mínimo permitido es 64 KB.
 - Está configurado en KB y el valor predeterminado lo administra dynatune.
 - Aumentar la memoria de trabajo a menudo ayuda a acelerar clasificación.
 - La configuración de la memoria de trabajo también se puede cambiar en base de sesión.

PERFORMANCE TUNING.

Connections Settings.

Maintenance work mem.

- Memoria máxima en KB a utilizar en mantenimiento operaciones como VACUUM, CREATE INDEX y ALTER TABLA AÑADIR CLAVE EXTRANJERA.
- = El valor mínimo permitido es 1024 KB.
- Está configurado en KB y el valor predeterminado lo administra dynatune.
- Rendimiento para aspirar y restaurar volcados de bases de datos se puede mejorar aumentando este valor.

PERFORMANCE TUNING.

Connections Settings.

Autovacuum work mem.

- Cantidad máxima de memoria a utilizar por cada proceso de trabajo de autovacuum.
- El valor predeterminado es -1, indica que mantenimiento work_mem para ser utilizado en su lugar.

PERFORMANCE TUNING.

Connections Settings.

effective cache size

- Tamaño de la memoria disponible para caché de disco que está disponible para un consulta única.
- = Un valor más alto favorece las exploraciones de índice.
- = Este parámetro no reserva caché de disco del kernel; solo se usa con fines de estimación.
 - “of RAM” es una configuración conservadora.
 - % de RAM es más agresivo.
 - Encuentre el valor óptimo mirando las estadísticas del sistema operativo después de aumentar o disminuyendo este parámetro.

PERFORMANCE TUNING.

Connections Settings.

temp_file_limit.

- Cantidad máxima de espacio en disco que una sesión puede usar para archivos temporales.
- Una transacción que intente exceder este límite será cancelado.
- El valor predeterminado es -1 (sin límite).
- Esta configuración restringe el espacio total utilizado en cualquier instante por todos los archivos temporales utilizados por una determinada sesión de Postgres.

PERFORMANCE TUNING.

Wal Parameters-Checkpoint.

Checkpoints.

- gs las páginas modificadas en memoria actuales (conocidas como páginas sucias).
- Se produce un punto de control automático cada vez que se alcanza el `max_wal_size * max_wal_size`.
- Distancia máxima entre puntos de control WAL automáticos.
- Cada segmento de archivo de registro es de 16 megabytes por defecto.
- Se fuerza un punto de control cuando se alcanza el `max_wal_size`.
- Una configuración más grande da como resultado menos puntos de control.
- El valor predeterminado es 1 GB.
- `max_wal_size` es un límite suave y el tamaño WAL puede exceder durante una carga.

PERFORMANCE TUNING.

Wal Parameters-Checkpoint.

Checkpoint_timeout.

- Tiempo máximo entre puntos de control WAL automáticos en segundos antes de que se fuerce un punto de control.
 - Una configuración más grande da como resultado menos puntos de control
 - El rango es de 30 a 3600 segundos.
- = El valor predeterminado es 300 segundos.

PERFORMANCE TUNING.

Postgres Expert.

Dashboard Properties SQL Statistics Dependencies Dependents Monitoring  Postgres Expert

Postgres Expert Report

Generated On: 2022-09-29 21:17:29 Go to: Postgres Enterprise Manager Server

Summary

Servers Tested: 1 Rules Checked: 31 High Alerts: 1 Medium Alerts: 3 Low Alerts: 2

Server: Postgres Enterprise Manager Server (127.0.0.1:5444)

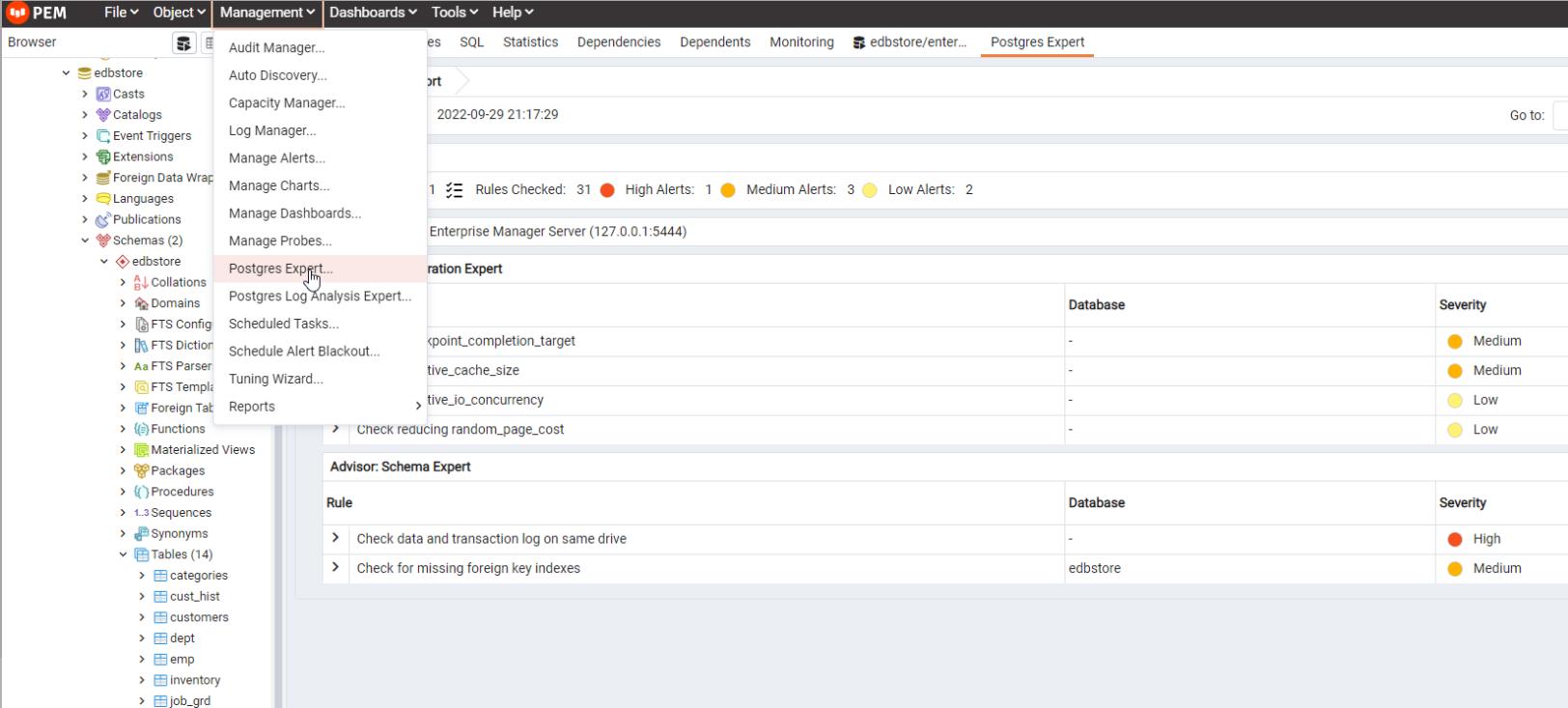
Advisor: Configuration Expert

Rule	Database	Severity
> Check checkpoint_completion_target	-	Medium
> Check effective_cache_size	-	Medium
> Check effective_io_concurrency	-	Low
> Check reducing random_page_cost	-	Low

Advisor: Schema Expert

Rule	Database	Severity
> Check data and transaction log on same drive	-	High
> Check for missing foreign key indexes	edbstore	Medium

Desde PEM corre un Postgres Expert.



The screenshot shows the PEM application interface. The top navigation bar includes File, Object, Management, Dashboards, Tools, and Help. The Management dropdown is open, showing options like Audit Manager..., Auto Discovery..., Capacity Manager..., Log Manager..., Manage Alerts..., Manage Charts..., Manage Dashboards..., Manage Probes..., Postgres Expert..., Postgres Log Analysis Expert..., Scheduled Tasks..., Schedule Alert Blackout..., Tuning Wizard..., Reports, and Advisor: Schema Expert. The 'Postgres Expert...' option is highlighted with a mouse cursor. The main content area displays a timeline from 2022-09-29 21:17:29, a summary of checked rules (31 total, 1 High Alert, 3 Medium Alerts, 2 Low Alerts), and two tables of tuning recommendations:

Rule	Database	Severity
Check point_completion_target	-	Medium
Check active_cache_size	-	Medium
Check active_ioConcurrency	-	Low
Check reducing random_page_cost	-	Low

Rule	Database	Severity
Check data and transaction log on same drive	-	High
Check for missing foreign key indexes	edbstore	Medium

PERFORMANCE TUNING.

Pg_test_fsync Tool.

El método wal_sync se utiliza para forzar las actualizaciones de WAL al salir al disco.

- pg_test_fsync puede determinar el más rápido.
- wal_sync_method en su sistema específico.
- pg_test_fsync informa de la operación promedio de sincronización de archivos.
- Información de diagnóstico para un problema de E/S identificado.

PERFORMANCE TUNING.

Pg_test_fsync Tool.

```
[enterprisedb@vm1 ~]$ pg_test_fsync
5 seconds per test
O_DIRECT supported on this platform for open_datasync and open_sync.

Compare file sync methods using one 8kB write:
(in wal_sync_method preference order, except fdatasync is Linux's default)
  open_datasync           10000.870 ops/sec    100 usecs/op
  fdatasync             9373.333 ops/sec    107 usecs/op
  fsync                  7823.123 ops/sec    128 usecs/op
  fsync_writethrough     n/a
  open_sync              8559.044 ops/sec    117 usecs/op

Compare file sync methods using two 8kB writes:
(in wal_sync_method preference order, except fdatasync is Linux's default)
  open_datasync           4734.306 ops/sec    211 usecs/op
  fdatasync               8367.599 ops/sec    120 usecs/op
  fsync                  7365.104 ops/sec    136 usecs/op
  fsync_writethrough     n/a
  open_sync              3764.238 ops/sec    266 usecs/op

Compare open_sync with different write sizes:
(This is designed to compare the cost of writing 16kB in different write
open_sync sizes.)
  1 * 16kB open_sync write      7347.938 ops/sec    136 usecs/op
  2 * 8kB open_sync writes    ■
```

PERFORMANCE TUNING.

Parallel Queries-Parallel Plans.

Parallel scans - Los siguientes tipos de tablas paralelas admiten escaneos:

- ❖ Parallel sequential scan: permite que multiple workers realicen una escaneo secuencial.
- ❖ Parallel bitmap heap scan: permite un solo escaneo de índice para enviar trabajadores paralelos para procesar diferentes áreas del montón.
- ❖ Parallel index scan or parallel index-only scan: permite el B-tree páginas de índice para ser buscadas por worker process separados.
- ❖ Parallel joins: permite bucles anidados, combinaciones hash o combinaciones realizarse en paralelo.

PERFORMANCE TUNING.

Parallel Queries-Parallel Plans.

EDB Postgres Advanced Server admite la ejecución paralela de consultas de solo lectura.

Se puede habilitar y configurar mediante la configuración.

Parámetros.

- max_parallel_workers_per_gather (predeterminado 2) - Habilita el análisis de consultas en paralelo.
- parallel_tuple_cost (predeterminado 0.1) - Costo estimado de transferir una tupla de un proceso de trabajo paralelo a otro proceso.
- Parallel_setup_cost (predeterminado 1000.0) - Estima el costo de lanzamiento en paralelo procesos de trabajo.

PERFORMANCE TUNING.

Parallel Queries-Parallel Plans.

EDB Postgres Advanced Server admite la ejecución paralela de consultas de solo lectura.

Se puede habilitar y configurar mediante la configuración.

Parámetros.

- min_parallel_table_scan_size (predeterminado 8 MB) - Establece la cantidad mínima de datos de la tabla que se deben escanear para que se considere un escaneo paralelo.
- min_parallel_index_scan_size (predeterminado 512 KB) - Establece la cantidad mínima de datos de índice que se deben escanear para que se considere un escaneo paralelo.
- force_parallel_mode (predeterminado desactivado) - Útil cuando se prueba el escaneo de consultas paralelas incluso cuando no hay beneficio de rendimiento.

PERFORMANCE TUNING.

Ejemplo Parallel Query Scan.

```
edb=# CREATE TABLE test (id int);
CREATE TABLE
edb=# INSERT INTO test VALUES(generate_series(1,100000000));
INSERT 0 100000000
edb=# ANALYZE test;
ANALYZE
```

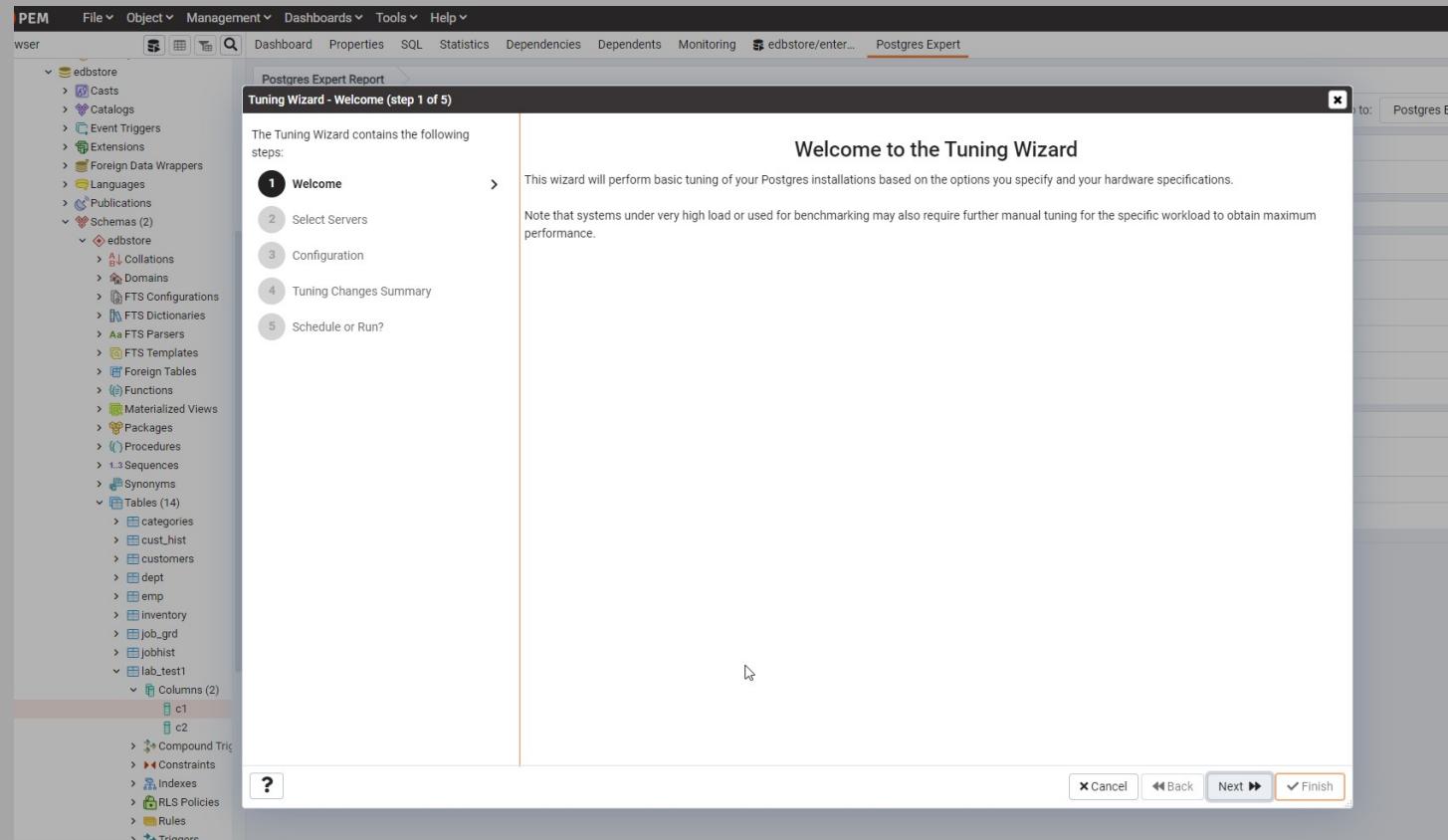
```
edb=# SHOW max_parallel_workers_per_gather;
max_parallel_workers_per_gather
-----
0
(1 row)

edb=# EXPLAIN ANALYZE SELECT * FROM test WHERE id = 1;
QUERY PLAN

-----
Seq Scan on test  (cost=0.00..1692478.40 rows=1 width=4) (actual time=2.480..44
814.208 rows=1 loops=1)
  Filter: (id = 1)
  Rows Removed by Filter: 99999999
  Planning time: 0.346 ms
  Execution time: 44814.247 ms
(5 rows)
```

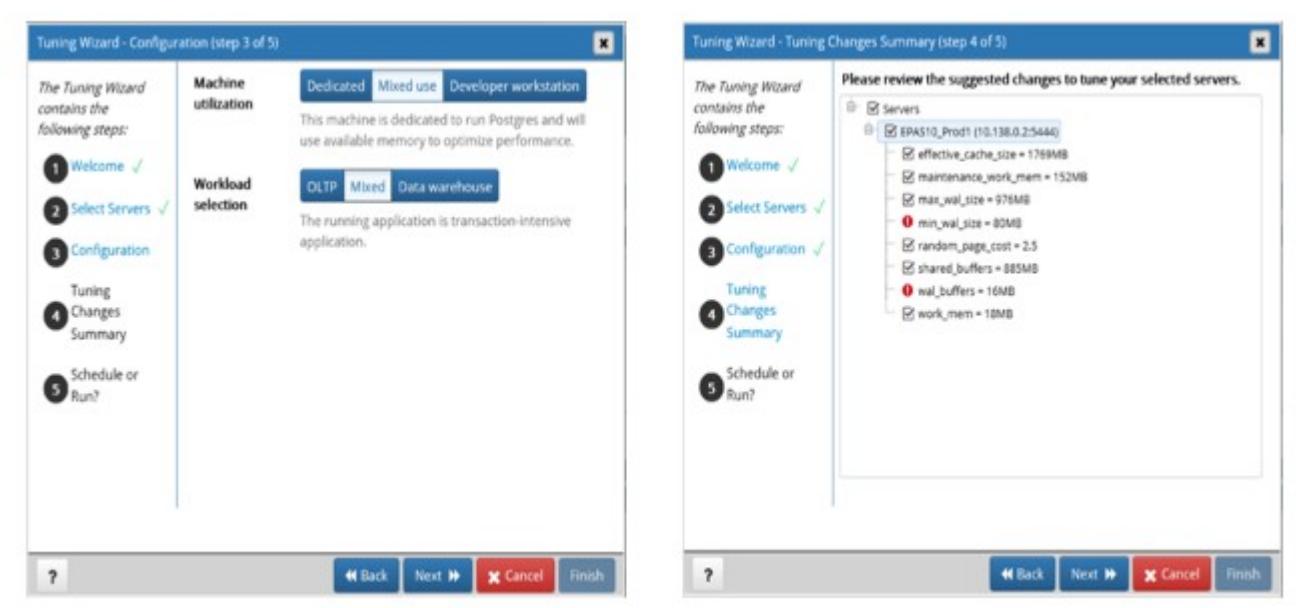
PERFORMANCE TUNING.

PEM Tuning Wizard.



PERFORMANCE TUNING.

PEM Tuning Wizard.



The Tuning Wizard contains the following steps:

- 1 Welcome ✓
- 2 Select Servers ✓
- 3 Configuration
- 4 Tuning Changes Summary
- 5 Schedule or Run?

Machine utilization
Dedicated Mixed use Developer workstation
This machine is dedicated to run Postgres and will use available memory to optimize performance.

Workload selection
DLTP Mixed Data warehouse
The running application is transaction-intensive application.

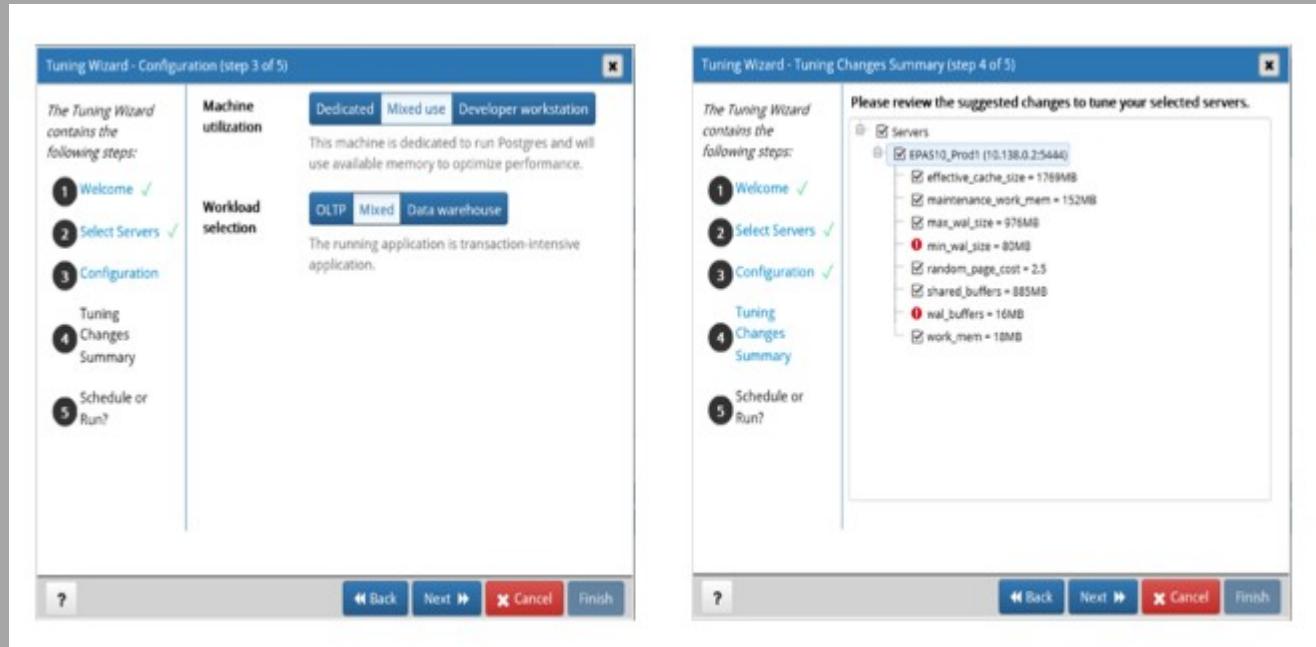
Please review the suggested changes to tune your selected servers.

Servers
EPAS10_Prod1 (10.138.0.2:5444)

- effective_cache_size = 1788MB
- maintenance_work_mem = 152MB
- max_wal_size = 976MB
- min_wal_size = 80MB
- random_page_cost = 2.5
- shared_buffers = 885MB
- wal_buffers = 16MB
- work_mem = 10MB

PERFORMANCE TUNING.

PEM Tuning Wizard.



Tuning Wizard - Configuration (step 3 of 5)

The Tuning Wizard contains the following steps:

- 1 Welcome ✓
- 2 Select Servers ✓
- 3 Configuration
- 4 Tuning Changes Summary
- 5 Schedule or Run?

Machine utilization
This machine is dedicated to run Postgres and will use available memory to optimize performance.

Workload selection
The running application is transaction-intensive application.

Tuning Wizard - Tuning Changes Summary (step 4 of 5)

Please review the suggested changes to tune your selected servers.

Servers: EPAS10_Prod1 (10.138.0.2:5444)

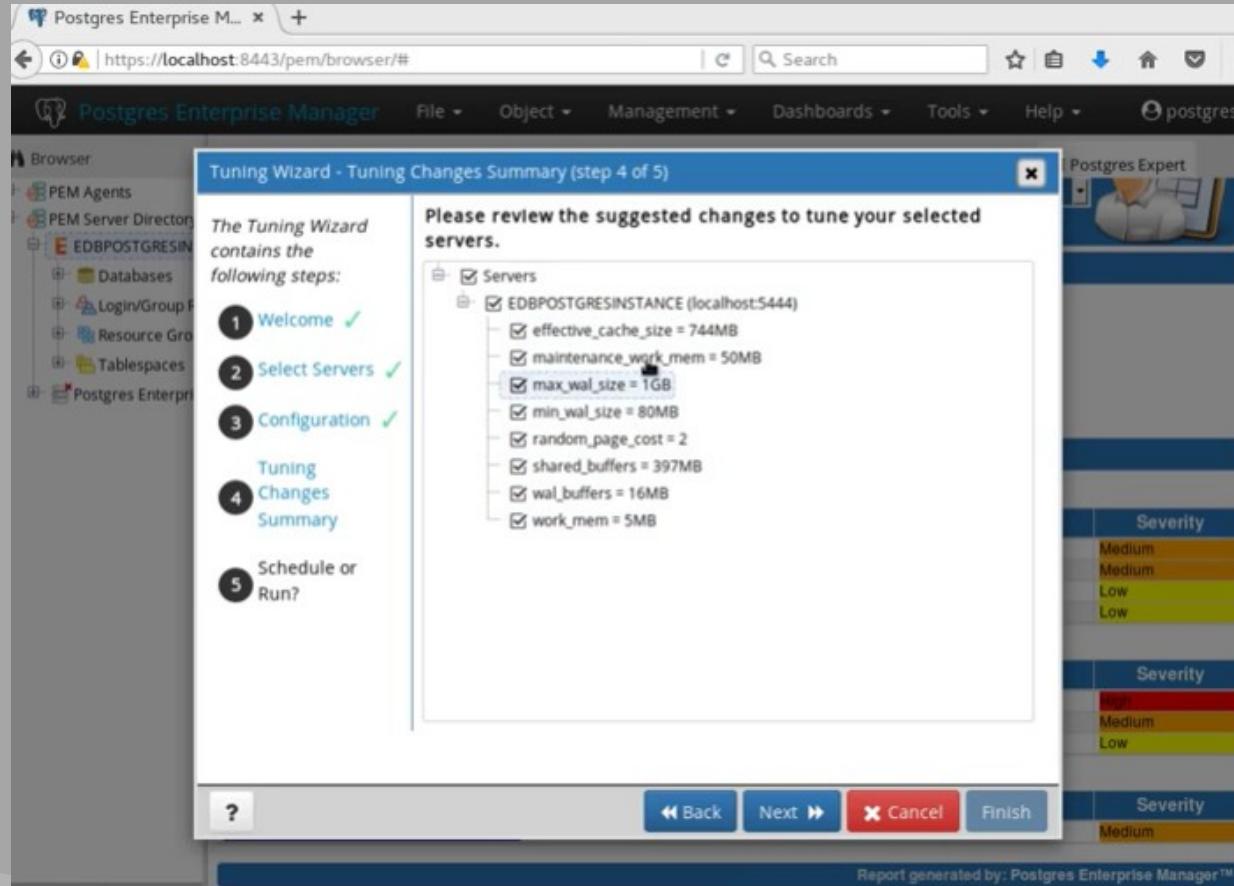
- effective_cache_size = 176MB
- maintenance_work_mem = 152MB
- max_wal_size = 976MB
- min_wal_size = 80MB**
- random_page_cost = 2.5
- shared_buffers = 885MB
- wal_buffers = 16MB**
- work_mem = 10MB

Configuration

- 1 Welcome ✓
- 2 Select Servers ✓
- 3 Configuration
- 4 Tuning Changes Summary
- 5 Schedule or Run?

PERFORMANCE TUNING.

PEM Tunning Wizard.



PERFORMANCE TUNING.

PEM Tuning Wizard.

Tuning Wizard Report

Generated: 2018-03-07 17:16:53 Jump to: EPAS10_Prod1



Summary

Settings	Value
Number of servers selected:	1
Machine utilization:	Mixed use
Workload profile:	Mixed

Server: EPAS10_Prod1 (10.138.0.2:5444)

GUC Parameter	Original Value	Recommended Value
effective_cache_size	4096MB	1769MB
maintenance_work_mem	88MB	152MB
max_wal_size	16GB	976MB
random_page_cost	4	2.5
shared_buffers	593MB	885MB
work_mem	4MB	18MB

Report generated by: Postgres Enterprise Manager™

PERFORMANCE TUNING.

DRITA.

Dynamic Runtime Instrumentation Tools Architecture (DRITA):

- Permite a los administradores de bases de datos consultar catálogos.
- Determina los eventos de espera que afectan el rendimiento.
- DRITA registra la información del evento de espera mientras consume recursos mínimos.
- Compara instantáneas para evaluar el rendimiento de un sistema.
- snapshots identificados por un número de identificación único.
- Funciones disponibles para crear, visualizar y comparar snapshots.
- DRITA consume recursos mínimos del sistema.

PERFORMANCE TUNING.

DRITA Snapshots.

Una instantánea o snapshot es un conjunto guardado de datos de rendimiento del sistema en un momento dado.

- Un número de identificación único identifica cada instantánea.
- Los números de ID de instantáneas se utilizan con los informes DRITA para devolver estadísticas de rendimiento del sistema.

PERFORMANCE TUNING.

DRITA Reports.

- Revise los cinco eventos principales en un informe dado
- Busque cualquier evento que tome un tiempo desproporcionado y utilice gran porcentaje de los recursos

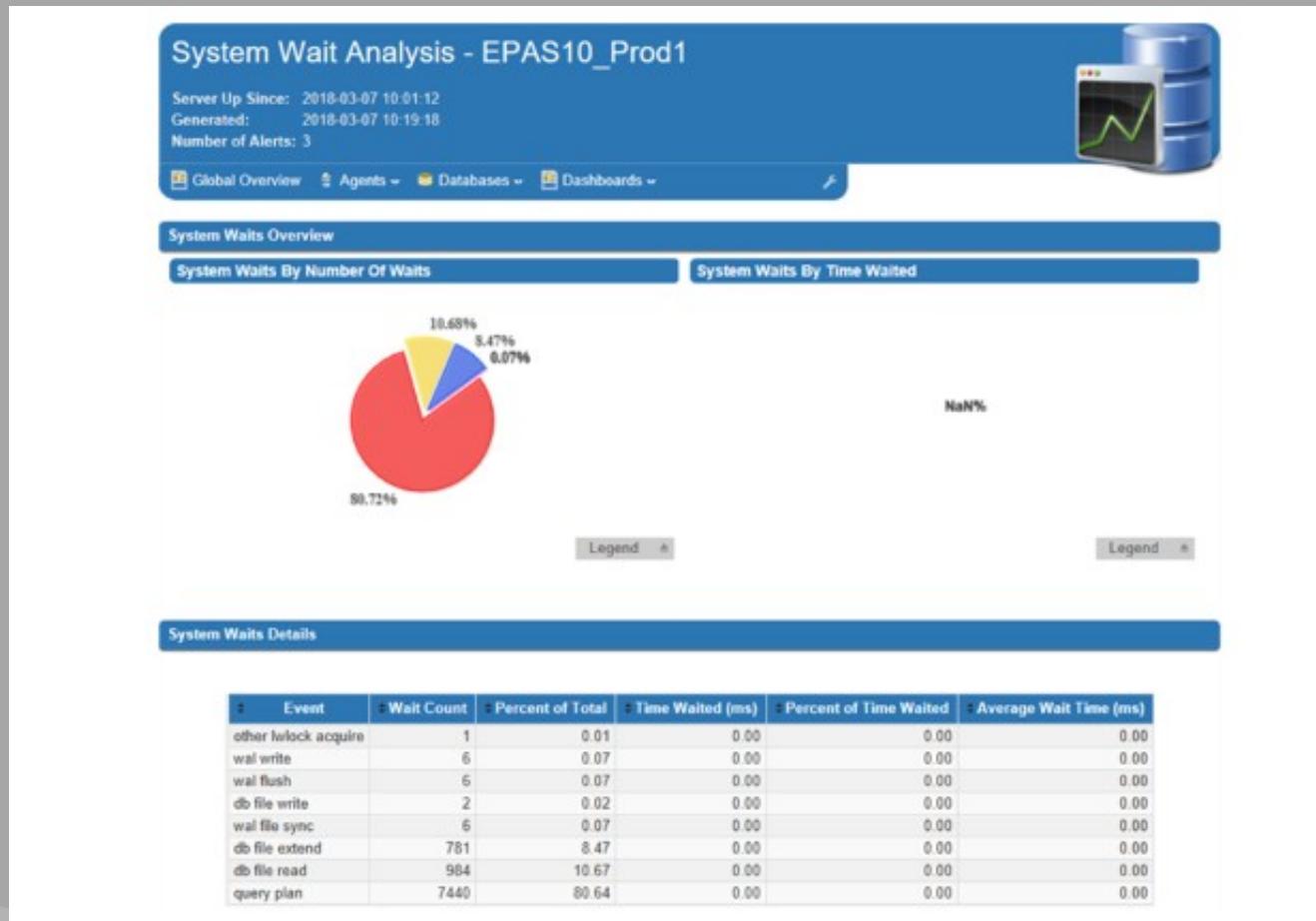
Las esperas deben evaluarse en el contexto de la CPU uso y tiempo total

Cuando evalúe eventos, esté atento a:

- Checkpoint waits.
- WAL-related waits.
- SQL parse waits.
- db file random reads.
- db file random writes.
- btree random lock acquires.

PERFORMANCE TUNING.

PEM – System Wait Analysis Dashboard.



PERFORMANCE TUNING.

SQL Profiler.

SQL Profiler es un asistente gráfico disponible en PEM.

Las capacidades de PEM SQL Profiler son muy similares a Microsoft SQL Server Profiler puede ser usado para:

- Capturar cargas de trabajo en un seguimiento de SQL.
- Supervisar y analizar consultas SQL.
- Diagnosticar consultas de ejecución lenta.
- = Ver las estadísticas de ejecución de consultas.
- Se puede programar un seguimiento de SQL para que se ejecute durante cargas de trabajo pesadas.
- Obtener asesoramiento sobre índices.

El complemento SQL Profiler debe estar instalado y configurado para cada base de datos Postgres.

PERFORMANCE TUNING.

Capacity Planning using Capacity Manager.

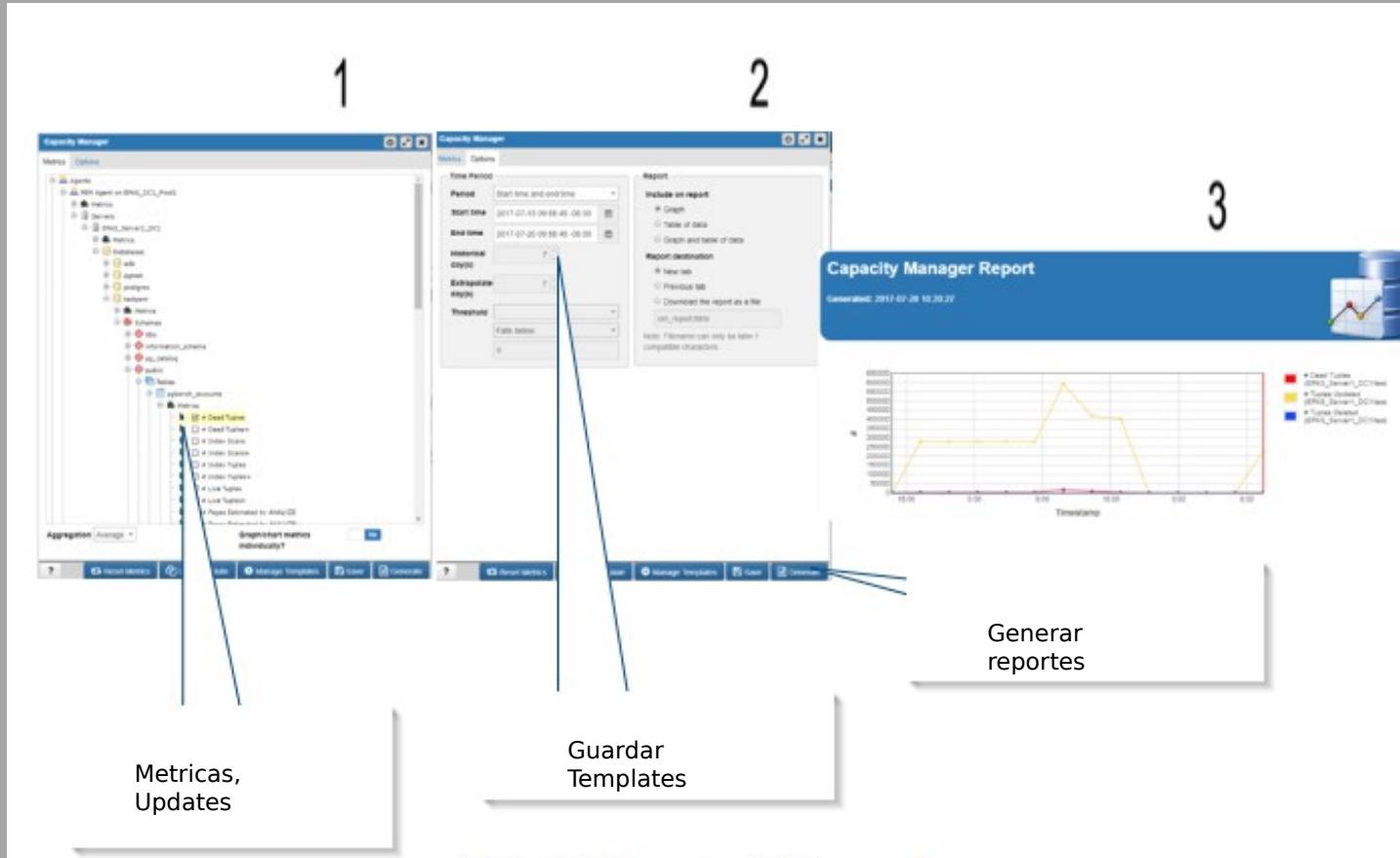
PEM contiene capacidades integradas para realizar planificación de la capacidad de la base de datos

La planificación de la capacidad ayuda al DBA al proporcionar respuestas a preguntas como:

- = ¿Cuánto almacenamiento necesitará mi base de datos dentro de 6 meses?
- = ¿Qué tan rápido está creciendo mi base de datos?
- = ¿Qué objetos son responsables del crecimiento de mi base de datos?
- = ¿Podrá mi servidor admitir otra instancia de base de datos?
- = ¿Está mejorando el rendimiento de mi base de datos, manteniéndose igual o cada vez peor?

PERFORMANCE TUNING.

Capacity Planning using Capacity Manager.



Tunea el parámetro en Postgres.conf.

- Cambie el número máximo de conexiones simultáneas en el clúster a 50.

PERFORMANCE BENCHMARKING



TOOLS Benchmarking.

- Herramientas open source de Benchmarking.
- Pgbench.
- BenchmarkSL.
- HammerDB.

PERFORMANCE BENCHMARKING

Innovación tecnológica en defensa y seguridad
DECSEF

TOOLS Benchmarking.

- pgbench

Programa simple para ejecutar pruebas comparativas en EDB Advanced Server.

No se requiere instalación.

- El ejecutable pgbench se encuentra por defecto en
 - /opt/edb/asl0/bin/

PERFORMANCE BENCHMARKING

Innovación tecnológica en defensa y seguridad
DECSEF

TOOLS Benchmarking.

- Pgbench Working
- Conecte múltiples sesiones de base de datos y ejecute el mismo SQL varias veces.
- Calcula la tasa de transacción promedio (transacciones por segundo).

La carga de trabajo predeterminada incluye cinco SELECT, UPDATE e INSERT comandos por transacción.

La prueba TPC-B requiere tablas específicas en una base de datos.

PERFORMANCE BENCHMARKING

Innovación tecnológica en defensa y seguridad
DECSEF

TOOLS Benchmarking.

```
- pgbench -i [ other-options ] dbname
[enterprisedb@localhost ~]$ createdb testdb
Password:
[enterprisedb@localhost ~]$ pgbench -i testdb
Password:
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
creating tables...
100000 of 100000 tuples (100%) done (elapsed 0.24 s, remaining 0.00 s)
vacuum...
set primary keys...
done.
```

PERFORMANCE BENCHMARKING

Innovación tecnológica en defensa y seguridad
DECSEF

TOOLS Benchmarking.

pgbench -i CREA CUATRO TABLAS pgbench_accounts, pgbench_branches, pgbench_history, y pgbench_tellers.

relname	reltuples
pgbench_accounts	100000
pgbench_branches	1
pgbench_history	0
pgbench_tellers	10
(4 rows)	

Incrementa el número de filas -s (scale factor).
Option.

F factor (default 100).

- unlogged-tables Crea tablas sin login.
- tablespace and --index-tablespace Puede ser usado en la inicialización.

PERFORMANCE BENCHMARKING



Innovación tecnológica en defensa y seguridad
DECSEF

Default Script.

```
=# BEGIN;
=# UPDATE pgbench_accounts SET abalance = abalance + :delta
WHERE
    aid = :aid;
=# SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
=# UPDATE pgbench_tellers SET tbalance = tbalance + :delta
WHERE tid
    = :tid;
=# UPDATE pgbench_branches SET bbalance = bbalance + :delta
WHERE
    bid = :bid;
=# INSERT INTO pgbench_history (tid, bid, aid, delta, mtime)
VALUES
    (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
=# END;
```

PERFORMANCE BENCHMARKING

Ejemplo pgbench.

```
[enterprisedb@localhost ~]$ pgbench -c 10 -T 60 testdb
Password:
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 1
query mode: simple
number of clients: 10
number of threads: 1
duration: 60 s
number of transactions actually processed: 53188
latency average = 11.287 ms
tps = 885.971896 (including connections establishing)
tps = 886.159912 (excluding connections establishing)
[enterprisedb@localhost ~]$
```

PERFORMANCE BENCHMARKING

Ejemplo pgbench.

```
[enterprisedb@vm1 ~]$ createdb edbtest
[enterprisedb@vm1 ~]$ pgbench -i -s 50 edbtest
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
creating tables...
100000 of 5000000 tuples (2%) done (elapsed 0.07 s, remaining 3.45 s)
200000 of 5000000 tuples (4%) done (elapsed 0.16 s, remaining 3.79 s)
300000 of 5000000 tuples (6%) done (elapsed 0.27 s, remaining 4.25 s)
400000 of 5000000 tuples (8%) done (elapsed 0.35 s, remaining 4.08 s)
500000 of 5000000 tuples (10%) done (elapsed 0.45 s, remaining 4.08 s)
600000 of 5000000 tuples (12%) done (elapsed 0.54 s, remaining 3.97 s)
700000 of 5000000 tuples (14%) done (elapsed 0.65 s, remaining 3.97 s)
800000 of 5000000 tuples (16%) done (elapsed 0.76 s, remaining 3.97 s)
```

```
[enterprisedb@vm1 ~]$ psql edbtest
psql.bin (10.1.5)
Type "help" for help.

edbtest=# \dt+
              List of relations
 Schema |      Name      | Type | Owner | Size | Description
-----+-----+-----+-----+-----+-----+
 public | pgbench_accounts | table | enterprisedb | 641 MB |
 public | pgbench_branches | table | enterprisedb | 40 kB |
 public | pgbench_history | table | enterprisedb | 0 bytes |
 public | pgbench_tellers | table | enterprisedb | 56 kB |
(4 rows)

edbtest=# \q
[enterprisedb@vm1 ~]$ pgbench -T 60 -c 50 edbtest
starting vacuum...end.
```

EXTENSIONS.

Extension Modules.

Los módulos en el directorio "Extension" son características adicionales para EDB Postgres.

- Generalmente, los módulos de extensión no están incluidos en el núcleo base de datos.
- Es posible que todavía estén en desarrollo.

También conocido como Módulos Adicionales Suministrados.

EXTENSIONS.

Extension Modules Example.

```
edb=# CREATE TABLE images (img_id NUMBER,img LO);
ERROR: type "lo" does not exist
LINE 1: CREATE TABLE images (img_id NUMBER,img LO);
          ^
edb=# CREATE EXTENSION lo;
CREATE EXTENSION
edb=# CREATE TABLE images (img_id NUMBER,img LO);
CREATE TABLE
edb=# INSERT INTO images values(1,lo_import('/home/enterprisedb/logo.png'));
INSERT 0 1
edb=# TABLE images;
   img_id |   img
-----+-----
      1 | 65596
(1 row)

edb=# SELECT lo_export(img,'/home/enterprisedb/logo.png') FROM images WHERE img_id=1;
   lo_export
-----
      1
(1 row)
```

EXTENSIONS.

Extension Modules Example.

```
edb=# CREATE TABLE images (img_id NUMBER,img LO);
ERROR: type "lo" does not exist
LINE 1: CREATE TABLE images (img_id NUMBER,img LO);
          ^
edb=# CREATE EXTENSION lo;
CREATE EXTENSION
edb=# CREATE TABLE images (img_id NUMBER,img LO);
CREATE TABLE
edb=# INSERT INTO images values(1,lo_import('/home/enterprisedb/logo.png'));
INSERT 0 1
edb=# TABLE images;
   img_id | img
-----+-----
      1 | 65596
(1 row)

edb=# SELECT lo_export(img,'/home/enterprisedb/logo.png') FROM images WHERE img_id=1;
   lo_export
-----
      1
(1 row)
```

TABLE PARTITIONING.

Partitioning.

- Hablamos de **particionamiento** al dividir lógicamente una gran tabla en piezas físicas más pequeñas. El particionamiento puede proporcionar varios beneficios:
 - Mejora en rendimiento de las consultas.
 - Aprovechar escaneo secuencial en lugar de un índice o de acceso aleatorio distribuido por toda la tabla al acceder a parte de una sola partición.
 - Las cargas o borrados masivos se pueden lograr agregando o eliminando particiones.
 - Los datos que se usan con poca frecuencia se pueden migrar a medios de almacenamiento más baratos o lentos.
- Los beneficios normalmente se logran en tablas muy grandes. Como regla general se mira que la tabla exceda la memoria física del servidor de la base de datos.

TABLE PARTITIONING.

Advanced Server soporta tres tipos de particionamiento.

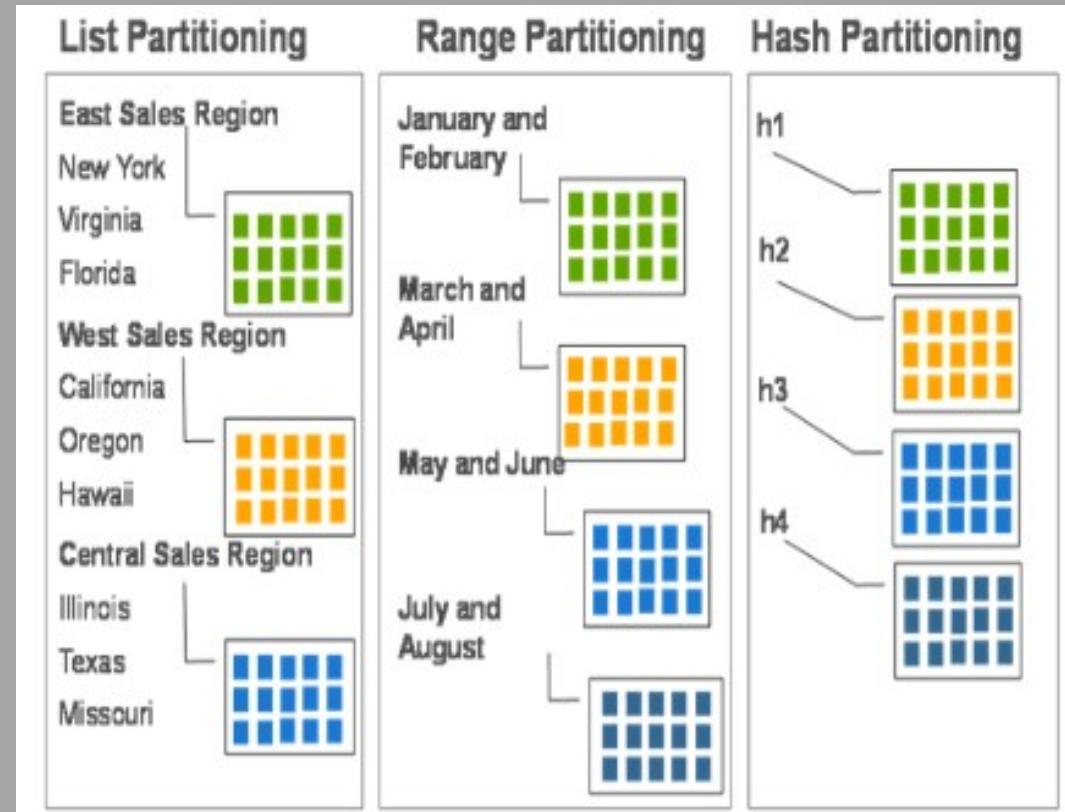


TABLE PARTITIONING.

Advanced Server soporta tres tipos de particionamiento.

- **Particionamiento por Rango:** la tabla es dividida por “rangos” definidos por una columna clave o un conjunto de columnas, sin superposición entre los rangos de valores asignados a diferentes particiones.
- **Particionamiento por Lista:** la tabla se divide enumerando explícitamente qué valores clave aparecen en cada partición.
- **Particionamiento por Hash:** Es un nuevo mecanismo de partición, para cuando no se puede decidir sobre un rango o partición de lista (porque no se está seguro de la cantidad de datos para cada partición).

TABLE PARTITIONING.

Sintaxis.

```
=# CREATE TABLE [schema.]table_name
    table_definition
    PARTITION BY LIST|RANGE|HASH(column)
    [SUBPARTITION BY {RANGE|LIST|HASH(column[, column
    ]....)}(list_partition_definition[,,
    list_partition_definition]...);

Where list_partition_definition is:
    PARTITION [partition_name]
    VALUES (value[, value]...) -NO VALUES FOR HASH PARTITION
    [TABLESPACE tablespace_name]
    [(subpartition, ...)]
```

TABLE PARTITIONING.

Fast Pruning.

- Fast Pruning ocurre temprano en el proceso de planificación.
Solo funciona en consultas con cláusulas WHERE o JOIN.

Controlado por un parámetro llamado:

poda edb_enable

- Reduce el número de particiones que el planificador debe considerar.
- No funciona en tablas subparticionadas o RANGE particiones divididas en más de una columna.

TABLE PARTITIONING.

Constarint Exclusion.

- La exclusión de restricciones ocurre tarde en la planificación del proceso.
- Se examinan las restricciones definidas para cada partición.

Controlado por la exclusión de la restriccción parámetro que debe establecerse en partición.

TABLE PARTITIONING.

Listing Partitioned Tables.

- ALL_TAB_PARTITIONS: Partición del usuario actual.
- ALL_TAB_SUBPARTITIONS: Sub particiones de todas las tablas.

edb=# \d ALL_TAB_PARTITIONS View "sys.all_tab_partitions"			edb=# \d ALL_TAB_SUBPARTITIONS View "sys.all_tab_subpartitions"		
Column	Type	Modifiers	Column	Type	Modifiers
table_owner	text		table_owner	text	
schema_name	text		schema_name	text	
table_name	text		table_name	text	
composite	text		partition_name	text	
partition_name	text		subpartition_name	text	
subpartition_count	bigint		high_value	text	
high_value	text		high_value_length	integer	
high_value_length	integer		subpartition_position	integer	
partition_position	integer		tablespace_name	text	
tablespace_name	text		pct_free	numeric	
pct_free	numeric		pct_used	numeric	
pct_used	numeric		ini_trans	numeric	
ini_trans	numeric		max_trans	numeric	
max_trans	numeric		initial_extent	numeric	
initial_extent	numeric		next_extent	numeric	

TABLE PARTITIONING.

Adding a New Partitioned Table.

- Use el comando ALTER TABLE.. ADD PARTITION para agregar una partición a una tabla particionada existente.

La nueva partición debe ser del mismo tipo.

- **Syntax:**

```
ALTER TABLE table_name ADD
PARTITION partition_definition;
Where partition_definition is
{list_partition | range_partition }
```

- **Example:**

```
=# ALTER TABLE sales ADD PARTITION east_asia VALUES
('CHINA', 'KOREA');
```

TABLE PARTITIONING.

Partitioned An Existing Table.

- The `ALTER TABLE...EXCHANGE PARTITION` command swaps an existing table with a partition or sub-partition
- Syntax of the `EXCHANGE PARTITION` command:

```
ALTER TABLE target_table
    EXCHANGE PARTITION|SUBPARTITION
    target_partition|
    target_subpartition
    WITH TABLE source_table [(WITH | WITHOUT)
    VALIDATION];
```
- Preparing a new partition is resource intensive
- Load the data into a staging table and use `EXCHANGE PARTITION`

TABLE PARTITIONING.

Partitioned Examples.

```
[enterprisedb@vm1 ~]$ psql edbstore edbuser
psql.bin (10.1.5)
Type "help" for help.

edbstore=> \dt+ test
           List of relations
 Schema | Name | Type  | Owner | Size | Description
-----+-----+-----+-----+-----+
 edbuser | test | table | edbuser | 249 MB |
(1 row)

edbstore=> \d test
          Table "edbuser.test"
 Column |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----+
 id    | numeric          |           |          |
 name  | character varying |           |          |
 city  | character varying |           |          |

edbstore=> explain analyze select * from test where city='CityA';
```

TABLE PARTITIONING.

Partitioned Examples.

```
edbstore=> explain analyze select * from test where city='CityA';
                                         QUERY PLAN

-----
Seq Scan on test  (cost=0.00..94348.00 rows=1008167 width=18) (actual time=1.
508..674.580 rows=1000000 loops=1)
  Filter: ((city)::text = 'CityA'::text)
  Rows Removed by Filter: 4000000
  Execution time: 706.432 ms
  Result  (cost=0.00..0.00 rows=0 width=0)                                I
    One-Time Filter: '===[ HYPOTHETICAL PLAN ]==='::text
      -> Index Scan using "<hypothetical-index>:3" on test  (cost=0.56..31212.48
          rows=1008167 width=18)
          Index Cond: ((city)::text = 'CityA'::text)
(8 rows)

edbstore=> ■
```

TABLE PARTITIONING.

Partitioned Examples.

```
Partition key: LIST (city)
Partitions: testpart_a FOR VALUES IN ('CityA'),
            testpart_b FOR VALUES IN ('CityB'),
            testpart_c FOR VALUES IN ('CityC'),
            testpart_d FOR VALUES IN ('CityD')
```

```
edbstore=> insert into testpart select * from test;
INSERT 0 5000000
edbstore=> \dt+ testpart*
```

List of relations					
Schema	Name	Type	Owner	Size	Description
edbuser	testpart	table	edbuser	8192 bytes	
edbuser	testpart_a	table	edbuser	50 MB	
edbuser	testpart_b	table	edbuser	100 MB	
edbuser	testpart_c	table	edbuser	50 MB	
edbuser	testpart_d	table	edbuser	50 MB	
(5 rows)					

```
edbstore=> alter table test rename to oldtest;
ALTER TABLE
edbstore=> alter table testpart rename to test;
ALTER TABLE
edbstore=> explain analyze select * from test where city='CityA';
                                         QUERY PLAN
-----
                                         Append  (cost=0.00..18870.00 rows=1000000 width=18) (actual time=0.006..177.2
93 rows=1000000 loops=1)
    ->  Seq Scan on testpart_a  (cost=0.00..18870.00 rows=1000000 width=18) (actual
       time=0.006..122.826 rows=1000000 loops=1)
        Filter: ((city)::text = 'CityA'::text)
        Execution time: 206.359 ms
(4 rows)

edbstore=> ■
```

```
edbstore=> explain analyze select * from oldtest where city='CityA';
                                         QUERY PLAN
-----
                                         Seq Scan on oldtest  (cost=0.00..94348.00 rows=1008167 width=18) (actual time
=0.615..806.157 rows=1000000 loops=1)
    Filter: ((city)::text = 'CityA'::text)
    Rows Removed by Filter: 4000000
    Execution time: 838.874 ms
Result  (cost=0.00..0.00 rows=0 width=0)
One-Time Filter: '===[ HYPOTHETICAL PLAN ]==='::text
    ->  Index Scan using "<hypothetical-index>:10" on oldtest  (cost=0.56..3121
2.48 rows=1008167 width=18)
        Index Cond: ((city)::text = 'CityA'::text)
(8 rows)

edbstore=> ■
```

TABLE PARTITIONING.

Move a Partition.

- Usar ALTER TABLE.. MOVE PARTITION

Comando para mover una partición o sub partición a otro tablespace

Sintaxis:

```
ALTER TABLE name
MOVE PARTITION name
TABLESPACE tablespace_name;
```

TABLE PARTITIONING.

DROP a Partition.

- Usar ALTER TABLE.. DROP PARTITION.
Comando para borrar una partición.

Sintaxis:

```
ALTER TABLE table_name DROP
PARTITION partition_name;
```

Ejemplo:

```
ALTER TABLE sales DROP PARTITION americas;
```

TABLE PARTITIONING.

DROPPING y TRUNCATE.

- DROP o TRUNCATE borrarán o truncarán las particiones.

USA:

ALTER TABLE ...DROP PARTITION

Ejemplo:

USA: ALTER TABLE ...TRUNCATE PARTITION

TABLE PARTITIONING.

Handling Stary Values.

Los valores “Stray” son filas que no cumplen ninguna reglas de partición.

Parámetros a ser usados : DEFAULT o MAXVALUE.

se puede utilizar para capturar esas filas.

La sintaxis de DEFAULT es:

- PARTITION [partition_name] VALUES (DEFAULT)

La sintaxis de una regla MAXVALUE es:

- PARTICIÓN [partition_name] VALUES LESS THAN (MAXVALUE)

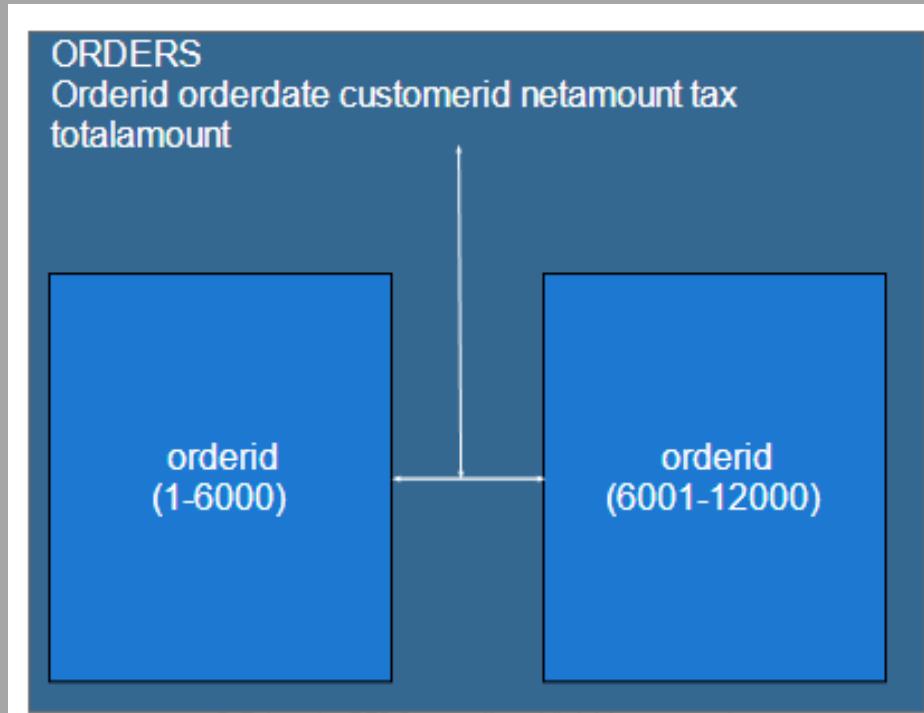
TABLE PARTITIONING.

Information Views.

Los valores “Stray” son filas que no cumplen ninguna reglas de partición.

- ALL_PART_TABLES.
- ALL_TAB_PARTITIONS.
- ALL_TAB_SUBPARTITIONS.
- ALL_PART_KEY_COLUMNS.
- ALL_SUBPART_KEY_COLUMNS.

- Observa la estructura de la table orders en la base edbstore y observa si esta particionada.
- La partición puedes hacerla así:



CONNECTION POOLING.

Pooling.

Los pool de conexiones son middleware que hablan el protocolo de la Base de Datos y las conexiones de Bases de Datos, para entre otras cosas aliviar el servidor de Base de Datos de mantener demasiados estados de clientes en memoria, por lo tanto las aplicaciones se conectan a estos pensando que es la Base de Datos.

- Sin un pool de conexiones, las conexiones de la Base de Datos son directamente manejadas por los procesos backend de PostgreSQL, un proceso por conexión.
- Hay dos productos que destacan por encima del resto: **PgBouncer** y **PgPool-II**:
 - Ambos productos están escritos en C.
 - Ambos productos están mantenidos muy activamente.
 - Ambos aumentan el rendimiento general para alto tráfico de conexiones.
 - PgBouncer es una herramienta liviana que permite reducir el tamaño

CONNECTION POOLING.

Pgpool II.

- Pgpool-II trabaja sobre Linux, Solaris,FreeBSD.
- pgpool-II Puede ser instalado en modo comando “yum.enterprisedb.com”.
- Pgpool-II Puede ser instalado en modo gráfico “ StackBuilder Plus” .

Los archivos de configuración predeterminados para pgpool-II son

- /opt/edb/pgpool3.6/etc/pgpool.conf y
- /opt/edb/pgpool3.6/etc/pep.conf

Hay varios modos de operación en pgpool-II

Cada modo tiene funciones asociadas que se pueden habilitar o deshabilitar para una configuración específica.

CONNECTION POOLING.

Pgpool II.

- Pgpool-II Tiene una interfaz de administración
- pep.conf se configura user/password para la autenticación
Despues de instalar: pgpool-II

/opt/edb/pgpool3.6/etc/pep.conf es creado.

```
$ cp /opt/edb/pgpool3.6/etc/pcp.conf.sample  
/opt/edb/pgpool3.6/etc/pcp.conf  
username:[password encrypted in md5]
```

Password debe de ser:

```
/opt/edb/pgpool3.6/bin/pg_md5 command:  
$ pg_md5 -p password: <your password>
```

CONNECTION POOLING.

Pgpool II.

pgpool.conf es el archivo de configuración principal de Pgpool-II.

- Cada modo de operación tiene parámetros de configuración específicos en pgpool.conf
- Hay cinco modos de ejecución diferentes en Pgpool-II para cada modo hay configuración de ejemplo en los siguientes archivos:

Operation mode	Configuration file name
Streaming replication mode	pgpool.conf.sample-stream
Replication mode	pgpool.conf.sample-replication
Master slave mode	pgpool.conf.sample-master-slave
Raw mode	pgpool.conf.sample
Logical replication mode	pgpool.conf.sample-logical

*Puedes copiar uno de ellos como pgpool.conf

CONNECTION POOLING.

Pgpool II Parameters.

Connections	• listen_addresses, port, pcp_port
Pools	• num_init_children, child_life_time, child_max_connections, client_idle_limit, enable_pool_hba, pool_passwd
Logs	• log_destination, log_connections, log_statement
File Location	• pid_file_name, logdir
Failovers	• failover_command, fallback_command, follow_master_command
Load Balancing	• replication_mode, master_slave_sub, load_balance_mode, replicate_select
Backends	• backend_hostname, backend_port, backend_weight, backend_data_directory, backend_flag

*Puedes copiar uno de los template pgpool.conf y ajustarlo a tus necesidades.

CONNECTION POOLING.

Pgpool II Parameters HBA.

Al igual que *pg_hba.conf* con Advanced Server, pgpool admite una autenticación de cliente similar.

• Esto se hace usando *pool_hba.conf*, de forma predeterminada, la autenticación *pool_hba* está deshabilitada

- Se debe de cambiar *enable_pool_hba* a "on" para habilitarlo.
- Se debe de copiar *pool_hba.conf.sample* como *pool_hba.conf* y edítelo si es necesario.

El formato y el uso son similares a *pg_hba.conf*.

Solo "trust", "reject", "md5" y "pam" para el campo MÉTODO son soportados

Para usar la autenticación md5, debe registrar el nombre de usuario y contraseña en el *pool_passwd*

*Puedes copiar uno de los template pgpool.conf y ajustarlo a tus necesidades.

CONNECTION POOLING.

Pgpool II START.

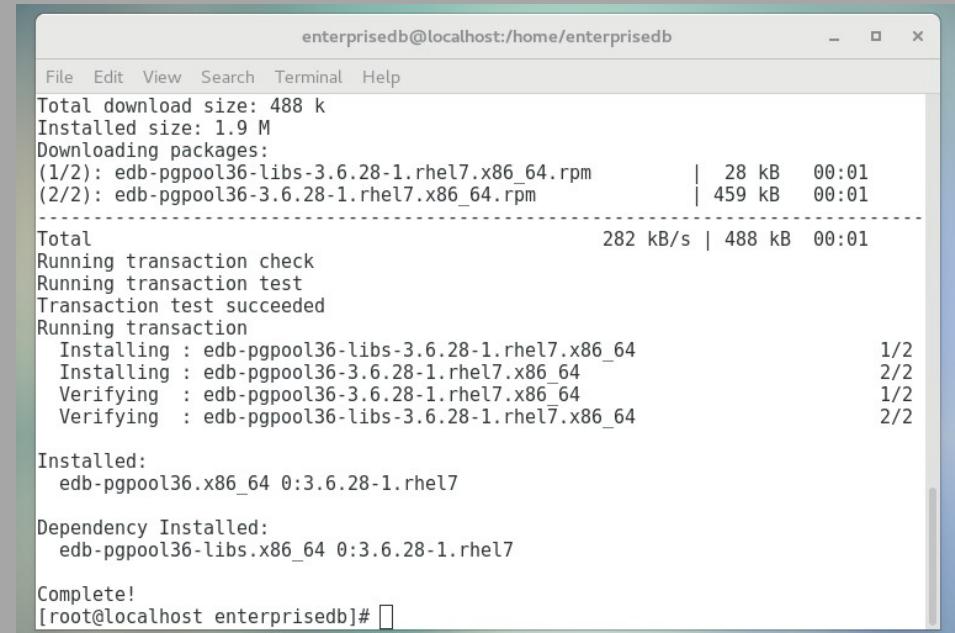
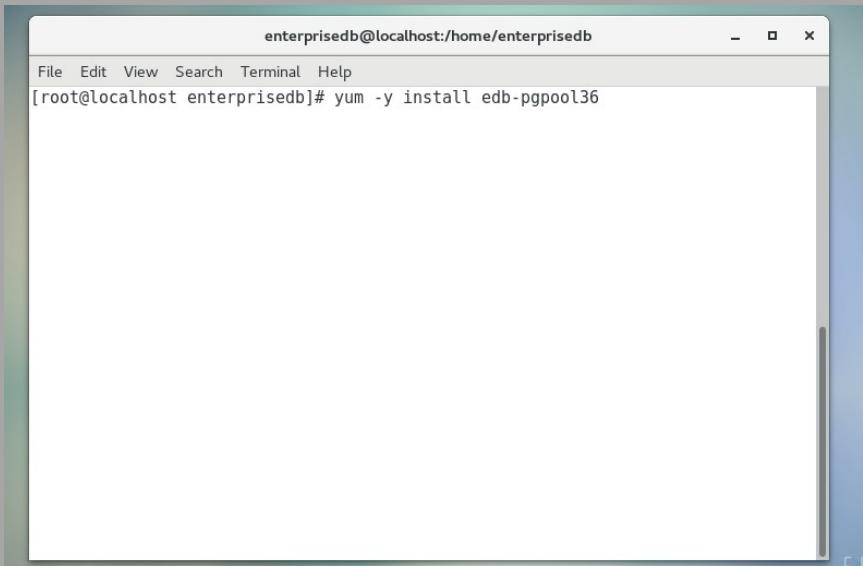
- Todos los backends y la base de datos del sistema (si es necesario) debe iniciarse antes de iniciar pgpool-II
- Se crea el servicio edb-pgpool-3.6 para iniciar/detener o comprobar el estado de pgpool-II

Sintaxis:

```
$ systemctl [ start | stop | status] edb-pgpool-3.6
```

CONNECTION POOLING.

Pgpool II Implementation.



```
enterprisedb@localhost:/home/enterprisedb
File Edit View Search Terminal Help
Total download size: 488 k
Installed size: 1.9 M
Downloading packages:
(1/2): edb-pgpool36-libs-3.6.28-1.rhel7.x86_64.rpm | 28 kB 00:01
(2/2): edb-pgpool36-3.6.28-1.rhel7.x86_64.rpm | 459 kB 00:01
Total 282 kB/s | 488 kB 00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : edb-pgpool36-libs-3.6.28-1.rhel7.x86_64 1/2
  Installing : edb-pgpool36-3.6.28-1.rhel7.x86_64 2/2
  Verifying : edb-pgpool36-3.6.28-1.rhel7.x86_64 1/2
  Verifying : edb-pgpool36-libs-3.6.28-1.rhel7.x86_64 2/2
Installed:
  edb-pgpool36.x86_64 0:3.6.28-1.rhel7
Dependency Installed:
  edb-pgpool36-libs.x86_64 0:3.6.28-1.rhel7
Complete!
[root@localhost enterprisedb]#
```

¿PREGUNTAS?





GRACIAS

- ✉ CONTACTO@DECSEF.COM
- 🌐 [HTTP://WWW.DECSEF.COM](http://WWW.DECSEF.COM)