



Prácticas Avanzado PostgreSQL

Índice de contenido

1	Introducción.....	3
2	Módulos.....	3
2.1	Preparación base de datos de prácticas.....	3
2.2	Módulo 1.....	3
2.2.1	Ejercicio 1.....	3
2.2.2	Solución 1.....	4
2.2.3	Ejercicio 2.....	5
2.2.4	Solución 2.....	5
2.2.5	Ejercicio 3.....	6
2.2.6	Solución 3.....	6
2.3	Módulo 2.....	6
2.3.1	Ejercicio 1.....	6
2.3.2	Solución 1.....	7
2.4	Módulo 3.....	8
2.4.1	Ejercicio 1.....	8
2.4.2	Solución 1.....	8
2.4.3	Ejercicio 2.....	9
2.4.4	Solución 2.....	9
2.4.5	Ejercicio 3.....	12
2.4.6	Solución 3.....	12
2.5	Módulo 4.....	13
2.5.1	Ejercicio 1.....	13
2.5.2	Solución 1.....	14
2.6	Módulo 5.....	16
2.6.1	Ejercicio 1.....	16
2.6.2	Solución 1.....	17
2.6.3	Ejercicio 2.....	18
2.6.4	Solución 2.....	18
2.7	Módulo 6.....	19
2.7.1	Ejercicio 1.....	19
2.7.2	Solución 1.....	19
2.8	Módulo 6.....	20
2.8.1	Ejercicio 1.....	20
	En este Lab aprenderemos a como utilizar PEM.....	20
2.8.2	Solución 1.....	20
	En este Lab aprenderemos a como utilizar PEM.....	20
2.8.3	Ejercicio 2.....	49
2.8.4	Solución 2.....	49
2.9	Módulo 8.....	51
2.9.1	Ejercicio 1.....	51
2.9.2	Solución 1.....	51

1 Introducción

El presente documento sirve de guía para la resolución de las prácticas que se encuentran en el curso avanzado de PostgreSQL.

2 Módulos

2.1 Preparación base de datos de prácticas

- Hopla software le proporcionará el material de prácticas del curso, y el script necesario para instalar la base de datos hoplastore. A continuación le detallamos los pasos:
 - Descargue del directorio compartido de dropbox, el fichero hoplastore.sql y colóquelo en un directorio accesible por el usuario postgres (o superusuario del servidor de base de datos).
 - Crear el usuario de base de datos hoplastore en el cluster disponible.
 - Crear la base de datos hoplastore usando como propietario el usuario hoplastore.
 - Conéctese en la base de datos hoplastore con el mismo usuario y cree el esquema hoplastore.
 - Desconéctese del psql.
 - Ejecute el comando psql con la opción -f para ejecutar el fichero hoplastore.sql que instalará una muestra de objetos usados en la formación.

2.2 Módulo 1

2.2.1 Ejercicio 1

- Previo estos laboratorios, usted debe haber creado la base de datos de pruebas hoplastore con algunos objetos.
- Habrá psql y:
 - Escriba una consulta que muestre el nombre del datafile en el que los la tabla *customers* están almacenados.
 - Escriba una query para encontrar el número de páginas y filas de la tabla *customers*.

Practicas Avanzado PostgreSQL

- Escriba una consulta que nos de el nombre del fichero de índice de la clave primaria de la tabla *customers*.
- Abra una terminal y muévase al directorio donde se sitúa el datafile de la tabla *customers*.

2.2.2 Solución 1

- **Previo estos laboratorios, usted debe haber creado la base de datos de pruebas hoplastore con algunos objetos.**
- **Habrà psql y:**
 - **Escriba una consulta que muestre el nombre del datafile en el que los la tabla *customers* están almacenados.**

```
hoplastore=> select relname, relfilenode from pg_class where relname = 'customers';
relname | relfilenode
-----+-----
customers |      24643
(1 fila)
```

- **Escriba una query para encontrar el número de páginas y filas de la tabla *customers*.**

```
hoplastore=> select relname,relpages,reltuples from pg_class where relname =
'customers';
relname | relpages | reltuples
-----+-----+-----
customers |      488 |    20000
(1 fila)
```

- **Escriba una consulta que nos de el nombre del fichero de índice de la clave primaria de la tabla *customers*.**

```
hoplastore=> select indexname, indexdef from pg_indexes where tablename =
'customers';
indexname | indexdef
-----+-----
customers_pkey | CREATE UNIQUE INDEX customers_pkey ON customers USING
btree (customerid)
ix_cust_username | CREATE UNIQUE INDEX ix_cust_username ON customers USING
btree (username)
(2 filas)
```

- **Abra una terminal y muévase al directorio donde se sitúa el datafile de la tabla *customers*.**

```
hoplastore=> \! find /opt/PostgreSQL/9.3/data -name 24643
/opt/PostgreSQL/9.3/data/base/24633/24643
```

```
hoplastore=> \! cd /opt/PostgreSQL/9.3/data/base/24633/
hoplastore=> \! pwd
/opt/PostgreSQL/9.3/data/base/24633/
```

2.2.3 Ejercicio 2

- Escriba un comando para encontrar los autovacuum workers que están actualmente ejecutándose en su servidor.
- Escriba un SQL statement para listar los autovacuum workers de su cluster.
- Encuentre un proceso con PID=10009 en su servidor consumiendo >80% de los recursos de CPU. Escriba un statement para encontrar este proceso en su cluster de base de datos.
- Escriba un SQL statement para proceder a matar este proceso.

2.2.4 Solución 2

- **Escriba un comando para encontrar los autovacuum workers que están actualmente ejecutándose en su servidor.**

```
-bash-4.1$ ps -fea | grep VACUUM
postgres 3461 3058 0 22:17 pts/0 00:00:00 grep VACUUM
```

- **Escriba un SQL statement para listar los autovacuum workers de su cluster.**

```
postgres=# select datname,pid,username,query from pg_stat_activity where query
like 'VACUUM%';
 datname | pid | username | query
-----+----+-----+-----
(0 filas)
```

- **Encuentre un proceso con PID=10009 en su servidor consumiendo >80% de los recursos de CPU. Escriba un statement para encontrar este proceso en su cluster de base de datos.**

```
postgres=# select datname,pid,username,query from pg_stat_activity where
pid=10009;
 datname | pid | username | query
-----+----+-----+-----
(0 filas)
```

- **Escriba un SQL statement para proceder a matar este proceso.**

```
postgres=# select pg_cancel_backend(10009);
postgres=# select pg_terminate_backend(10009);
```

2.2.5 Ejercicio 3

- Escriba una consulta para mostrar la tercera fila de la tercera columna de la tabla *orders*.
- Escriba una consulta para mostrar todas las filas físicamente ubicadas antes de la página tercera de la fila tercera de la tabla *orders*.

2.2.6 Solución 3

- **Escriba una consulta para mostrar la tercera fila de la tercera columna de la tabla *orders*.**

```
hoplastore=> select ctid, * from orders where ctid = '(2,3)';
ctid | orderid | orderdate | customerid | netamount | tax | totalamount
-----+-----+-----+-----+-----+-----+-----
(2,3) | 244 | 2004-01-07 | 5311 | 373.33 | 30.80 | 404.13
(1 fil
```

- **Escriba una consulta para mostrar todas las filas físicamente ubicadas antes de la página tercera de la fila tercera de la tabla *orders*.**

```
hoplastore=> select ctid, * from orders where ctid < '(2,3)';
ctid | orderid | orderdate | customerid | netamount | tax | totalamount
-----+-----+-----+-----+-----+-----+-----
(0,1) | 1 | 2004-01-27 | 7888 | 313.24 | 25.84 | 339.08
(0,2) | 2 | 2004-01-01 | 4858 | 54.90 | 4.53 | 59.43
(0,3) | 3 | 2004-01-17 | 15399 | 160.10 | 13.21 | 173.31
.....
```

2.3 Módulo 2

2.3.1 Ejercicio 1

- Simula el comportamiento del MVCC usando un ejemplo sobre la tabla *pdoducts*.
 - Abre un psql en un terminal (sesión 1).
 - Escriba una consulta para ver el precio total de los productos con *prod_id* entre 1 y 10.
 - Abra otro psql en otro terminal (sesión 2).
 - Comience una transacción explícita en el segundo psql. Escriba una consulta para actualizar el precio de todos los productos con *prod_id* entre 1 y 10 para que tenga un coste 10\$ superior al actual. Escriba una consulta para ver el precio total de los productos con *prod_id* entre 1 y 10. Este valor debe haber cambiado. NOTA: No haga *commit* de esta transacción.

Practicas Avanzado PostgreSQL

- Vuelva el psql 1. Escriba una consulta para ver el precio total de los productos con *prod_id* entre 1 y 10. Debe ver el valor antiguo.
- Vuelva al psql 2 y haga *commit*.
- Vuelva al psql 1. Escriba una consulta para ver el precio total de los productos con *prod_id* entre 1 y 10. Debe ver el valor nuevo.

2.3.2 Solución 1

- **Simula el comportamiento del MVCC usando un ejemplo sobre la tabla *pdoducts*.**
 - **Abre un psql en un terminal (sesión 1).**
 - **Escriba una consulta para ver el precio total de los productos con *prod_id* entre 1 y 10.**

```
hoplastore=> select sum(price) from products where prod_id between 1 and 10;
sum
-----
182.90
(1 row)
```

- **Abra otro psql en otro terminal (sesión 2).**
- **Comience una transacción explícita en el segundo psql. Escriba una consulta para actualizar el precio de todos los productos con *prod_id* entre 1 y 10 para que tenga un coste 10\$ superior al actual. Escriba una consulta para ver el precio total de los productos con *prod_id* entre 1 y 10. Este valor debe haber cambiado. NOTA: No haga *commit* de esta transacción.**

```
hoplastore=> begin;
BEGIN
hoplastore=> update products set price=price+10 where prod_id between 1 and 10;
UPDATE 10
hoplastore=> select sum(price) from products where prod_id between 1 and 10;
sum
-----
282.90
(1 row)
```

- **Vuelva el psql 1. Escriba una consulta para ver el precio total de los productos con *prod_id* entre 1 y 10. Debe ver el valor antiguo.**

```
hoplastore=> select sum(price) from products where prod_id between 1 and 10;
sum
-----
182.90
```



```
(1 row)
```

- **Vuelva al psql 2 y haga *commit*.**

```
hoplastore=> commit ;  
COMMIT
```

- **Vuelva al psql 1. Escriba una consulta para ver el precio total de los productos con *prod_id* entre 1 y 10. Debe ver el valor nuevo.**

```
hoplastore=> select sum(price) from products where prod_id between 1 and 10;  
sum  
-----  
282.90  
(1 row)
```

2.4Módulo 3

2.4.1 Ejercicio 1

- Los usuarios están preocupados por el bajo rendimiento de la base de datos *hoplastore*.
 - Configura el `postgresql.conf` para un óptimo performance basado en el tamaño de la base de datos *hoplastore*, la tabla más grande y demás información recolectada de la base de datos *hoplastore*.

2.4.2 Solución 1

- **Los usuarios están preocupados por el bajo rendimiento de la base de datos *hoplastore*.**
 - **Configura el `postgresql.conf` para un óptimo performance basado en el tamaño de la base de datos *hoplastore*, la tabla más grande y demás información recolectada de la base de datos *hoplastore*.**

```
# Modificar los parámetros de postgresql.conf de la siguiente manera:
```

```
Shared_buffers = 128MB  
Work_mem=2MB  
Maintenance_work_mem=64MB  
Effective_cache_size=512MB  
Max_connections=50  
Save and close the file.  
#Reiniciar el cluster  
postgres@student-VirtualBox:~$ service postgresql-9.4 restart  
Restarting PostgreSQL 9.4:  
Contraseña:  
waiting for server to shut down.... done
```



```
server stopped
waiting for server to start.... done
server started
touch: no se puede efectuar `touch' sobre «/var/lock/subsys/postgresql-9.4»: No
existe el archivo o el directorio
PostgreSQL 9.4 restarted successfully
```

2.4.3 Ejercicio 2

- Chequee el uso de los índices para todos los índices de usuario en la base de datos *hoplastore*.
- Reindexe todos los índices.
- Manualmente actualice las estadísticas para todos los objetos en la base de datos *hoplastore*.
- Verifique si se ha recolectado estadísticas de manera automática para las tablas de usuario de la base de datos *hoplastore*.

2.4.4 Solución 2

- **Chequee el uso de los índices para todos los índices de usuario en la base de datos *hoplastore*.**

```
hoplastore=> select * from pg_stat_user_indexes;
 relid | indexrelid | schemaname | relname | indexrelname | idx_scan |
idx_tup_read | idx_tup_fetch
-----+-----+-----+-----+-----+-----+-----
+-----+
16665 | 16733 | hoplastore | categories | categories_pkey | 0 | 0 |
0
16673 | 16735 | hoplastore | customers | customers_pkey | 0 | 0 |
| 0
16681 | 16737 | hoplastore | dept | dept_dname_uq | 0 | 0 |
0
16681 | 16739 | hoplastore | dept | dept_pk | 0 | 0 |
0
16684 | 16741 | hoplastore | emp | emp_pk | 0 | 0 |
0
16688 | 16743 | hoplastore | inventory | inventory_pkey | 0 | 0 |
0
16697 | 16745 | hoplastore | jobhist | jobhist_pk | 0 | 0 |
0
16712 | 16747 | hoplastore | orders | orders_pkey | 0 | 0 |
0
16717 | 16749 | hoplastore | products | products_pkey | 0 | 0 |
0
16670 | 16751 | hoplastore | cust_hist | ix_cust_hist_customerid | 0 | 0 |
| 0
16673 | 16752 | hoplastore | customers | ix_cust_username | 0 | 0 |
```

```
|      0
16712 |      16753 | hoplastore | orders   | ix_order_custid |      0 |      0 |
0
16709 |      16754 | hoplastore | orderlines | ix_orderlines_orderid |      0 |      0 |
0
16717 |      16755 | hoplastore | products | ix_prod_category |      0 |      0 |
0
16717 |      16756 | hoplastore | products | ix_prod_special  |      0 |      0 |
0
(15 rows)
```

- **Reindexe todos los índices.**

```
hoplastore=> reindex database hoplastore ;
NOTICE: table "pg_catalog.pg_class" was reindexed
NOTICE: table "pg_catalog.pg_statistic" was reindexed
NOTICE: table "pg_catalog.pg_type" was reindexed
NOTICE: table "hoplastore.categories" was reindexed
NOTICE: table "pg_catalog.pg_authid" was reindexed
NOTICE: table "pg_catalog.pg_attribute" was reindexed
NOTICE: table "pg_catalog.pg_proc" was reindexed
NOTICE: table "hoplastore.inventory" was reindexed
NOTICE: table "pg_catalog.pg_user_mapping" was reindexed
NOTICE: table "pg_catalog.pg_attrdef" was reindexed
NOTICE: table "pg_catalog.pg_constraint" was reindexed
NOTICE: table "pg_catalog.pg_index" was reindexed
NOTICE: table "pg_catalog.pg_operator" was reindexed
NOTICE: table "pg_catalog.pg_opfamily" was reindexed
NOTICE: table "pg_catalog.pg_opclass" was reindexed
NOTICE: table "pg_catalog.pg_am" was reindexed
NOTICE: table "pg_catalog.pg_amop" was reindexed
NOTICE: table "pg_catalog.pg_amproc" was reindexed
NOTICE: table "pg_catalog.pg_language" was reindexed
NOTICE: table "pg_catalog.pg_database" was reindexed
NOTICE: table "pg_catalog.pg_aggregate" was reindexed
NOTICE: table "pg_catalog.pg_rewrite" was reindexed
NOTICE: table "pg_catalog.pg_trigger" was reindexed
NOTICE: table "pg_catalog.pg_event_trigger" was reindexed
NOTICE: table "pg_catalog.pg_description" was reindexed
NOTICE: table "pg_catalog.pg_cast" was reindexed
NOTICE: table "pg_catalog.pg_enum" was reindexed
NOTICE: table "pg_catalog.pg_namespace" was reindexed
NOTICE: table "pg_catalog.pg_conversion" was reindexed
NOTICE: table "pg_catalog.pg_depend" was reindexed
NOTICE: table "pg_catalog.pg_db_role_setting" was reindexed
NOTICE: table "pg_catalog.pg_tablespace" was reindexed
NOTICE: table "pg_catalog.pg_pltemplate" was reindexed
NOTICE: table "pg_catalog.pg_auth_members" was reindexed
NOTICE: table "pg_catalog.pg_shdepend" was reindexed
NOTICE: table "pg_catalog.pg_shdescription" was reindexed
NOTICE: table "pg_catalog.pg_ts_config" was reindexed
```

```
NOTICE: table "pg_catalog.pg_ts_config_map" was reindexed
NOTICE: table "pg_catalog.pg_ts_dict" was reindexed
NOTICE: table "pg_catalog.pg_ts_parser" was reindexed
NOTICE: table "pg_catalog.pg_ts_template" was reindexed
NOTICE: table "pg_catalog.pg_extension" was reindexed
NOTICE: table "pg_catalog.pg_foreign_data_wrapper" was reindexed
NOTICE: table "pg_catalog.pg_foreign_server" was reindexed
NOTICE: table "pg_catalog.pg_foreign_table" was reindexed
NOTICE: table "pg_catalog.pg_default_acl" was reindexed
NOTICE: table "pg_catalog.pg_seclabel" was reindexed
NOTICE: table "pg_catalog.pg_shseclabel" was reindexed
NOTICE: table "pg_catalog.pg_collation" was reindexed
NOTICE: table "pg_catalog.pg_range" was reindexed
NOTICE: table "pg_catalog.pg_largeobject" was reindexed
NOTICE: table "hoplastore.job_grd" was reindexed
NOTICE: table "information_schema.sql_implementation_info" was reindexed
NOTICE: table "information_schema.sql_languages" was reindexed
NOTICE: table "information_schema.sql_packages" was reindexed
NOTICE: table "information_schema.sql_sizing" was reindexed
NOTICE: table "information_schema.sql_sizing_profiles" was reindexed
NOTICE: table "hoplastore.locations" was reindexed
NOTICE: table "hoplastore.products" was reindexed
NOTICE: table "pg_catalog.pg_largeobject_metadata" was reindexed
NOTICE: table "pg_catalog.pg_inherits" was reindexed
NOTICE: table "information_schema.sql_features" was reindexed
NOTICE: table "information_schema.sql_parts" was reindexed
NOTICE: table "hoplastore.cust_hist" was reindexed
NOTICE: table "hoplastore.customers" was reindexed
NOTICE: table "hoplastore.orders" was reindexed
NOTICE: table "hoplastore.orderlines" was reindexed
NOTICE: table "hoplastore.dept" was reindexed
NOTICE: table "hoplastore.emp" was reindexed
NOTICE: table "hoplastore.jobhist" was reindexed
REINDEX
```

- **Manualmente actualice las estadísticas para todos los objetos en la base de datos *hoplastore*.**

```
hoplastore=> analyze ;
WARNING: skipping "pg_authid" --- only superuser can analyze it
WARNING: skipping "pg_database" --- only superuser can analyze it
WARNING: skipping "pg_db_role_setting" --- only superuser can analyze it
WARNING: skipping "pg_tablespace" --- only superuser can analyze it
WARNING: skipping "pg_pltemplate" --- only superuser can analyze it
WARNING: skipping "pg_auth_members" --- only superuser can analyze it
WARNING: skipping "pg_shdepend" --- only superuser can analyze it
WARNING: skipping "pg_shdescription" --- only superuser can analyze it
WARNING: skipping "pg_shseclabel" --- only superuser can analyze it
ANALYZE
```

- **Verifique si se ha recolectado estadísticas de manera automática para**

las tablas de usuario de la base de datos *hoplastore*.

```
hoplastore=> select relname,last_autoanalyze,analyze_count from
pg_stat_user_tables where analyze_count<=0;
relname | last_autoanalyze | analyze_count
-----+-----
(0 rows)
```

2.4.5 Ejercicio 3

- Tome un full backup de la base de datos *hoplastore* empleando *pg_dump*.
- Elimine la base de datos *hoplastore* con un *drop database*.
- Cree una base de datos *hoplastore* en blanco.
- Verifique que *fsync* está a ON y compruebe el tiempo que tarda la base de datos *hoplastore* en restaurarse.
- Elimine la base de datos *hoplastore* con un *drop database* y cree de nuevo la base de datos *hoplastore*.
- Verifique que *fsync* está a OFF y compruebe el tiempo que tarda la base de datos *hoplastore* en restaurarse.

2.4.6 Solución 3

- **Tome un full backup de la base de datos *hoplastore* empleando *pg_dump*.**

```
root@student-VirtualBox:~# mkdir /pgbackup
root@student-VirtualBox:~# chown postgres:postgres /pgbackup/
root@student-VirtualBox:~# su - postgres
postgres@student-VirtualBox:~$ pg_dump -f
/pgbackup/hoplastore_advan_module3.sql -U postgres hoplastore
```

- **Elimine la base de datos *hoplastore* con un *drop database*.**

```
postgres@student-VirtualBox:~$ dropdb -U postgres hoplastore
```

- **Cree una base de datos *hoplastore* en blanco.**

```
postgres@student-VirtualBox:~$ createdb -O hoplastore hoplastore
```

- **Verifique que *fsync* está a ON y compruebe el tiempo que tarda la base de datos *hoplastore* en restaurarse.**

```
postgres@student-VirtualBox:~$ psql -c "show fsync;"
fsync
-----
on
```

```
(1 row)
postgres@student-VirtualBox:~$ time psql -f
/pgbackup/hoplastore_advan_module3.sql hoplastore hoplastore
real    0m4.948s
user    0m0.022s
sys     0m0.012s
```

- **Elimine la base de datos *hoplastore* con un *drop database* y cree de nuevo la base de datos *hoplastore*.**

```
postgres@student-VirtualBox:~$ dropdb -U postgres hoplastore
postgres@student-VirtualBox:~$ createdb -O hoplastore hoplastore
```

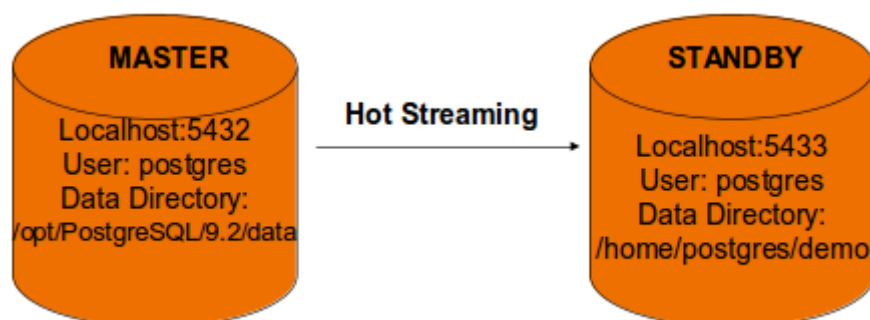
- **Verifique que *fsync* está a OFF y compruebe el tiempo que tarda la base de datos *hoplastore* en restaurarse.**

```
postgres@student-VirtualBox:~$ psql -c "show fsync;"
fsync
-----
off
(1 row)
postgres@student-VirtualBox:~$ time psql -f
/pgbackup/hoplastore_advan_module3.sql hoplastore hoplastore
real    0m3.631s
user    0m0.016s
sys     0m0.017s
```

2.5 Módulo 4

2.5.1 Ejercicio 1

- Implementa replicación mediante streaming replication entre dos clusters funcionando en tu servidor como se muestra:



2.5.2 Solución 1

- **Implementa replicación mediante streaming replication entre dos clusters funcionando en tu servidor como se muestra:**

```
# Loguearse como usuario que levanta el servicio
student@student-VirtualBox:~$ su - postgres
Contraseña:

# BackUp del fichero de configuración
postgres@student-VirtualBox:~$ cp /pgdata/cluster_test941/postgresql.conf
/pgdata/cluster_test941/postgresql.conf_modulo4

# Modificamos los siguientes parámetros
postgres@student-VirtualBox:~$ vi /pgdata/cluster_test941/postgresql.conf

wal_level = hot_standby
archive_mode = on
archive_command = 'cp %p /pgdata/arch/%f'
max_wal_senders = 3
wal_keep_segments = 20

# Guardar el fichero de configuración

# Crear la ruta de archivado
postgres@student-VirtualBox:~$ mkdir -p /pgdata/arch
postgres@student-VirtualBox:~$ ls -lth /pgdata/
total 8,0K
drwxr-xr-x  2 postgres postgrp  4,0K feb 23 20:09 arch

# Modificamos los accesos a la base de datos
postgres@student-VirtualBox:~$ vi /pgdata/cluster_test941/pg_hba.conf
host    replication    postgres    127.0.0.1/32          trust

# Reiniciar el servidor primario
postgres@student-VirtualBox:~$ service postgresql-9.4 restart
Restarting PostgreSQL 9.4:
Contraseña:
waiting for server to shut down.... done
server stopped
waiting for server to start.... done
server started
PostgreSQL 9.4 restarted successfully

# Se hace una copia del cluster
postgres@student-VirtualBox:~$ pg_basebackup -h localhost -U postgres -D
/pgdata/cluster_test941_replica
NOTICE: pg_stop_backup complete, all required WAL segments have been archived

# Se modifica el fichero postgresql.conf del esclavo. NOTA: El puerto no sería
necesario si la máquina no fuera la misma.
```

Practicar Avanzado PostgreSQL

```
port = 5433
hot_standby = on

# Se borra el postmaster.pid
postgres@student-VirtualBox:~$ rm /pgdata/cluster_test941_replica/postmaster.pid

# Se crea el fichero recovery.conf
standby_mode = 'on'
primary_conninfo = 'host=localhost port=5432'
trigger_file = '/tmp/trigger_hot_standby_failover'

# Se arranca el nodo esclavo en modo recuperación continua.
postgres@student-VirtualBox:~$ pg_ctl -D /pgdata/cluster_test941_replica/ start
server starting
postgres@student-VirtualBox:~$ 2016-02-23 20:21:43 CET LOG: redirecting log
output to logging collector process
2016-02-23 20:21:43 CET HINT: Future log output will appear in directory "pg_log".

# Se comprueba que ambos servicios están arriba.
postgres@student-VirtualBox:~$ ps -fea | grep -e "-D"
root      786    1  0 19:55 ?        00:00:00 /usr/sbin/sshd -D
postgres 3062  1816  0 20:12 ?        00:00:00 /opt/PostgreSQL941/bin/postgres -D
/pgdata/cluster_test941
postgres 3503  1816  0 20:25 pts/23   00:00:00 /opt/PostgreSQL941/bin/postgres
-D /pgdata/cluster_test941_replica
postgres 3514  3223  0 20:25 pts/23   00:00:00 grep -e -D

# Se comprueba la replicación
# Nodo master
postgres=# select * from test;
 id
----
  1
(1 row)

# Nodo 2
postgres=# select * from test;
 id
----
  1
(1 row)

# Se inserta en el nodo master
postgres=# insert into test values (2);
INSERT 0 1

# Se evisa el nodo standby
postgres=# select * from test;
 id
----
  1
  2
```


Practicas Avanzado PostgreSQL

(2 rows)

En el nodo standby se comprueba que no se puede insertar.

postgres=# insert into test values (3;

ERROR: cannot execute INSERT in a read-only transaction

Si se mata el nodo master, se puede promocionar el standby de la siguiente manera:

postgres@student-VirtualBox:~\$ pg_ctl -D /pgdata/cluster_test941_replica/ promote
server promoting

postgres@student-VirtualBox:~\$ psql -p 5433

Password:

psql.bin (9.4.1)

Type "help" for help.

No entry for terminal type "xterm";

using dumb terminal settings.

postgres=# insert into test values (3);

INSERT 0 1

postgres=# select * from test;

id

1

2

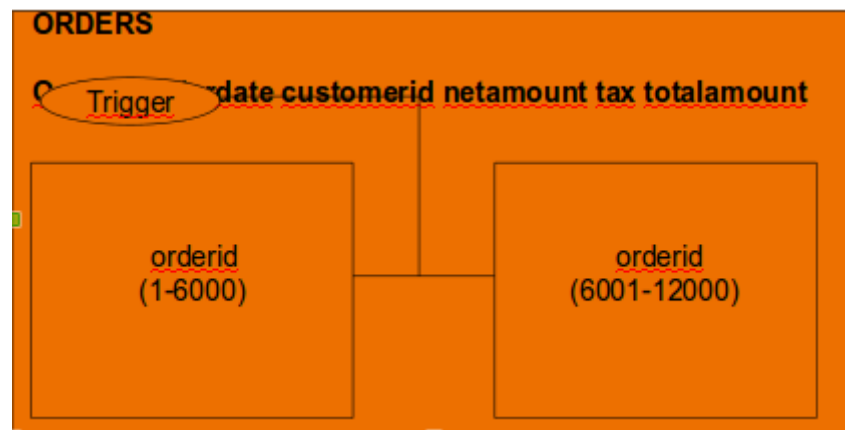
3

(4 rows)

2.6Módulo 5

2.6.1 Ejercicio 1

- Ver la estructura de la tabla *orders* en la base de datos de hoplastore
- Particionar la tabla según el esquema.



2.6.2 Solución 1

- Ver la estructura de la tabla *orders* en la base de datos de hoplastore
- Particionar la tabla según el esquema.

```
# Se crean las tablas
create table orders1 (like orders);
CREATE TABLE orders1_1 (CHECK (orderid >=1 and orderid <=6000)) INHERITS
(orders1);
CREATE TABLE orders1_2 (CHECK (orderid >=6001 and orderid <=12000)) INHERITS
(orders1);

# Se crean los índices
CREATE INDEX idx_orders1_1 ON orders1_1 (orderid);
CREATE INDEX idx_orders1_2 ON orders1_2 (orderid);

# Se crea la función del trigger
CREATE OR REPLACE FUNCTION orders1_part_func () RETURNS
trigger AS $$
BEGIN
IF NEW.orderid >= 1 and NEW.orderid <= 6000 then
INSERT INTO orders1_1 VALUES
(NEW.orderid,NEW.orderdate,NEW.customerid, NEW.netamount, NEW.tax,
NEW.totalamount);
ELSEIF NEW.orderid >= 6001 and NEW.orderid <= 12000 then
INSERT INTO orders1_2 VALUES
(NEW.orderid,NEW.orderdate,NEW.customerid, NEW.netamount, NEW.tax,
NEW.totalamount);
END IF;
RETURN NULL;
END; $$ language PLPGSQL;

# Se crea el trigger
CREATE TRIGGER orders1_part_trig
BEFORE INSERT ON orders1
FOR EACH ROW
EXECUTE PROCEDURE orders1_part_func();

# Se cargan los datos de orders sobre orders1
insert into orders1 select * from orders;

# Demostración de que se han cargado en una de las tablas hijas
hoplastore=> select count(*) from orders1_1 ;
count
-----
6000
(1 row)
```

2.6.3 Ejercicio 2

- Crear una clave primaria en la columna de cada tabla particionada.
- Ver el plan de ejecución de la siguiente consulta
 - `select * from orders1 where orderid=3000;`
- Desactivar el `constraint_exclusion` y ver el mismo plan de ejecución que en el paso anterior.

2.6.4 Solución 2

- **Crear una clave primaria en la columna de cada tabla particionada.**

```
hoplastore=> alter table orders1_1 add primary key(orderid);
ALTER TABLE
hoplastore=> alter table orders1_2 add primary key(orderid);
ALTER TABLE
```

- **Ver el plan de ejecución de la siguiente consulta**
 - `select * from orders1 where orderid=3000;`

```
hoplastore=> explain select * from orders1 where orderid = 3000;
               QUERY PLAN

-----
Append (cost=0.00..8.30 rows=2 width=45)
-> Seq Scan on orders1 (cost=0.00..0.00 rows=1 width=60)
    Filter: (orderid = 3000)
-> Index Scan using orders1_1_pkey on orders1_1 (cost=0.28..8.30 rows=1 width=30)
    Index Cond: (orderid = 3000)
(5 rows)
```

- **Desactivar el `constraint_exclusion` y ver el mismo plan de ejecución que en el paso anterior.**

```
hoplastore=> set constraint_exclusion = off;
SET
hoplastore=> explain select * from orders1 where orderid = 3000;
               QUERY PLAN

-----
Append (cost=0.00..16.60 rows=3 width=40)
-> Seq Scan on orders1 (cost=0.00..0.00 rows=1 width=60)
    Filter: (orderid = 3000)
```

```
-> Index Scan using orders1_1_pkey on orders1_1 (cost=0.28..8.30 rows=1 width=30)
      Index Cond: (orderid = 3000)
-> Index Scan using orders1_2_pkey on orders1_2 (cost=0.28..8.30 rows=1 width=30)
      Index Cond: (orderid = 3000)
(7 rows)
```

2.7 Módulo 6

2.7.1 Ejercicio 1

- Configurar pgbouncer para servir de pool de conexiones a su instancia.

2.7.2 Solución 1

- **Configurar pgbouncer para servir de pool de conexiones a su instancia.**

Pasos:

- Abrir la aplicación Stack Builder en el menú de aplicaciones
- Seleccionar cluster por defecto ejecutándolo en el puerto 5432 y clic next
- Selecciona pgbouncer de complementos y haga click en siguiente
- Seleccione el puerto (por defecto 6432) para pgbouncer y click next para instalar
- PgBouncer comenzará automáticamente y podrá ser parado y reiniciado utilizando pgbouncer service, que lo registrará automáticamente.

```
vi /opt/PgBouncer/share/pgbouncer.ini

hoplastore = host=127.0.0.1 port=5432 user=postgres password=postgres
dbname=hoplastore

root@student-VirtualBox:~# vi /opt/PgBouncer/etc/userlist.txt
"postgres" "postgres"

root@student-VirtualBox:~# /etc/init.d/pgbouncer restart
2016-02-24 09:16:06.374 3134 LOG File descriptor limit: 1024 (H:4096),
max_client_conn: 100, max_fds possible: 150

postgres@student-VirtualBox:~$ psql -p 6432 hoplastore postgres
Password for user postgres:
psql.bin (9.4.1)
Type "help" for help.

No entry for terminal type "xterm";
```

using dumb terminal settings.

2.8Módulo 6

2.8.1 Ejercicio 1

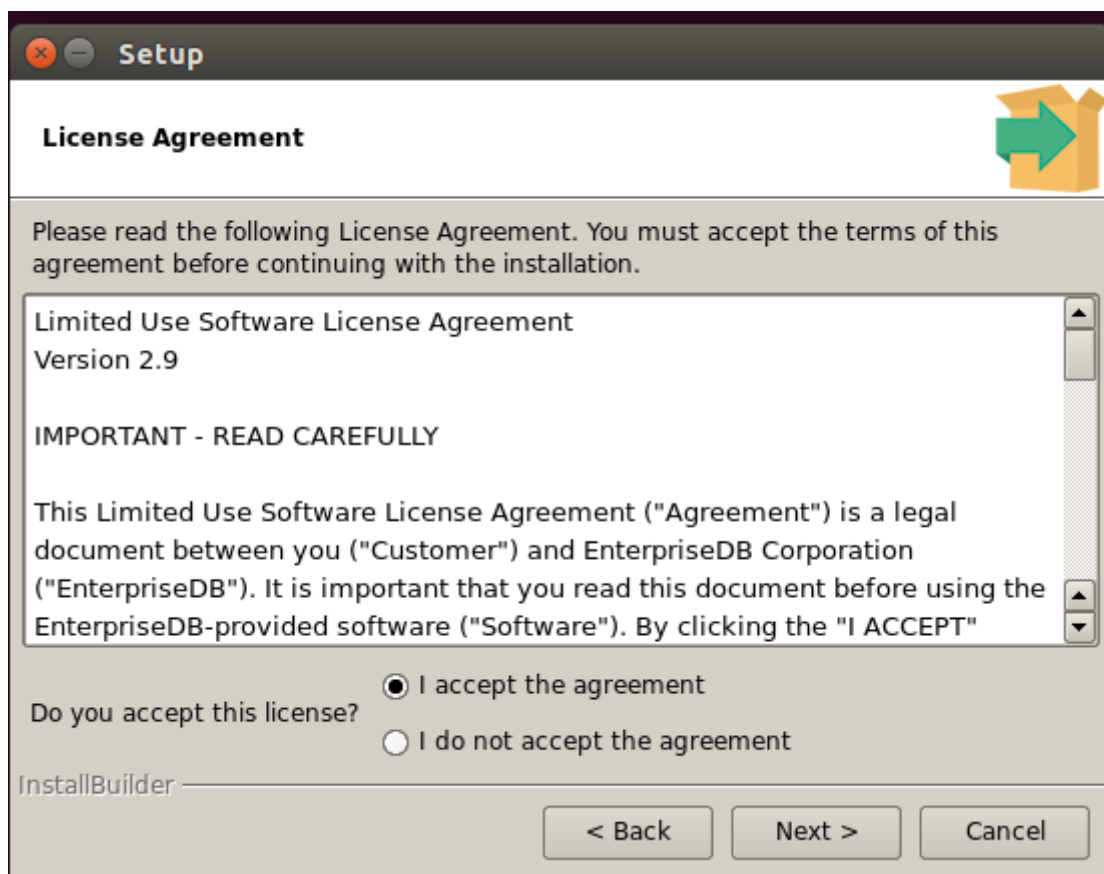
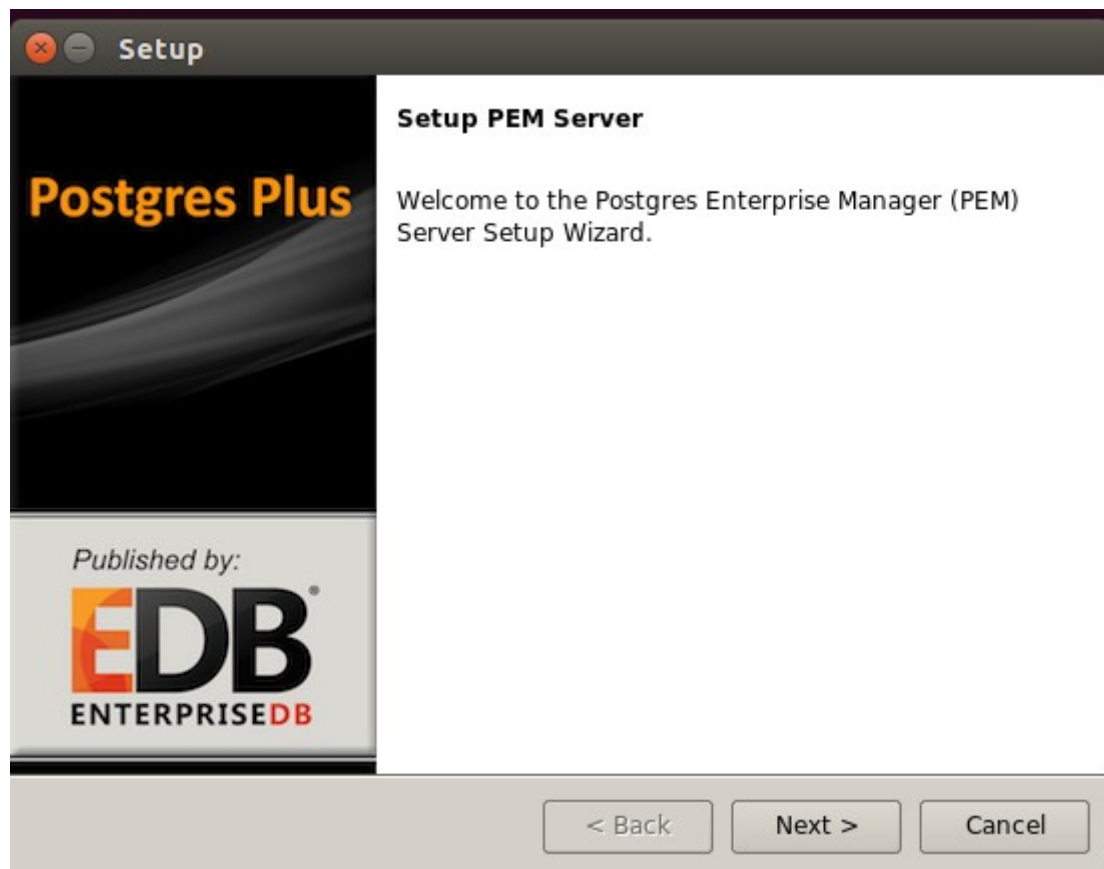
En este Lab aprenderemos a como utilizar PEM

- Descargar e instalar Postgres Enterprise Manager Server
- Descargar e instalar Postgres Enterprise Manager Client
- Abrir PEM Client
- El Cluster predeterminado enlazado se ejecuta en el puerto 5432 con PEM agent para coleccionar la monitorización de datos
- Mirar panel de control global para PEM server
- Mirar la actividad actual de la base de datos del panel de control para su cluster predeterminado
- Ver el panel de control de la memoria para su cluster predeterminado
- Mirar información OS para el servidor donde el cluster predeterminado se esté ejecutando

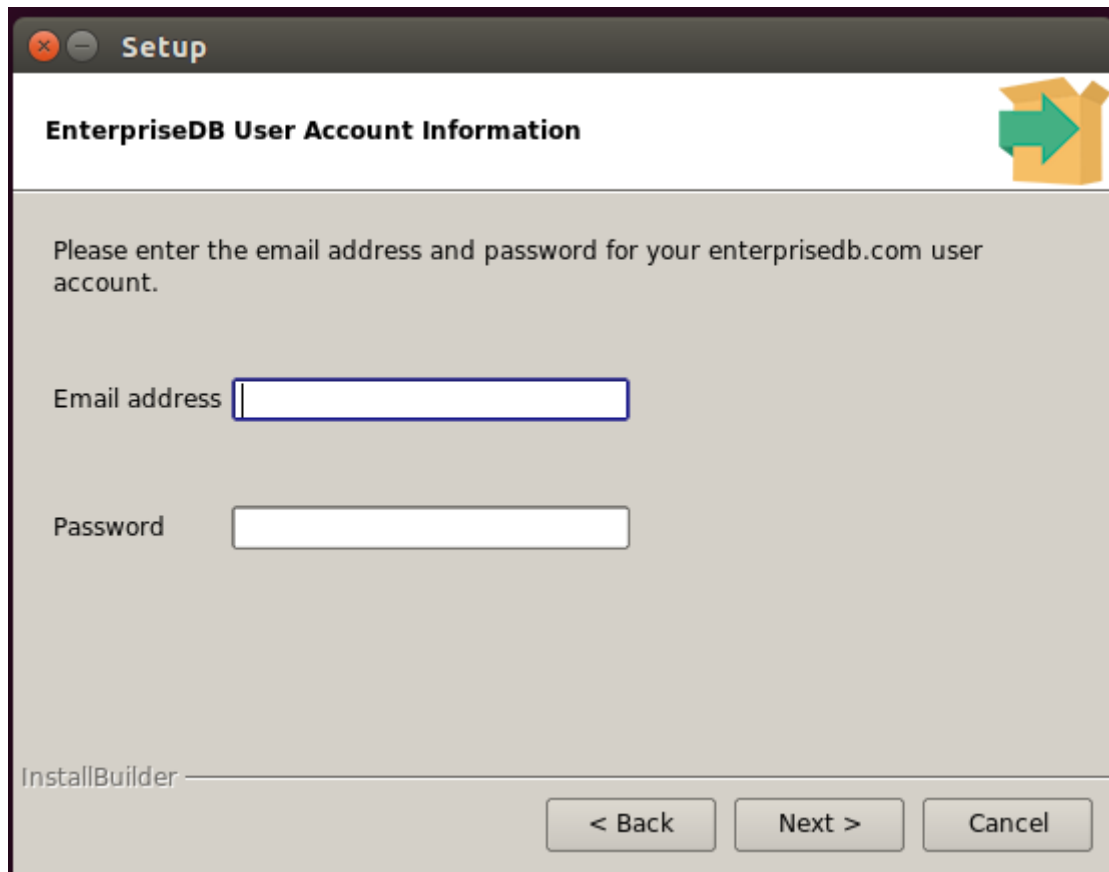
2.8.2 Solución 1

En este Lab aprenderemos a como utilizar PEM

- **Descargar e instalar Postgres Enterprise Manager Server**







The image shows a window titled "Setup" with a standard macOS-style title bar (red, yellow, and green buttons). The window's main content area is titled "EnterpriseDB User Account Information" in bold black text. To the right of this title is a yellow box with a green arrow pointing right. Below the title, the text reads: "Please enter the email address and password for your enterprisedb.com user account." There are two input fields: "Email address" and "Password". The "Email address" field is currently selected, indicated by a blue border. At the bottom left of the window, the text "InstallBuilder" is visible. At the bottom right, there are three buttons: "< Back", "Next >", and "Cancel".

Setup

EnterpriseDB User Account Information

Please enter the email address and password for your enterprisedb.com user account.

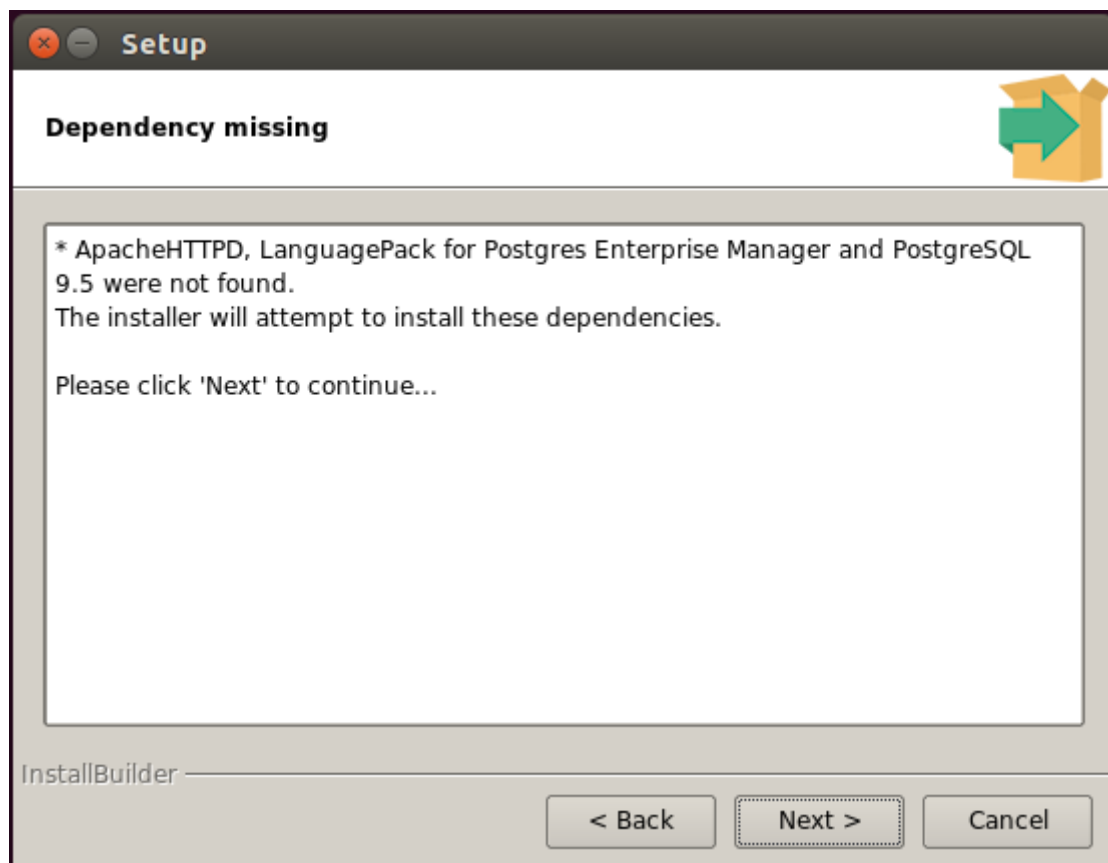
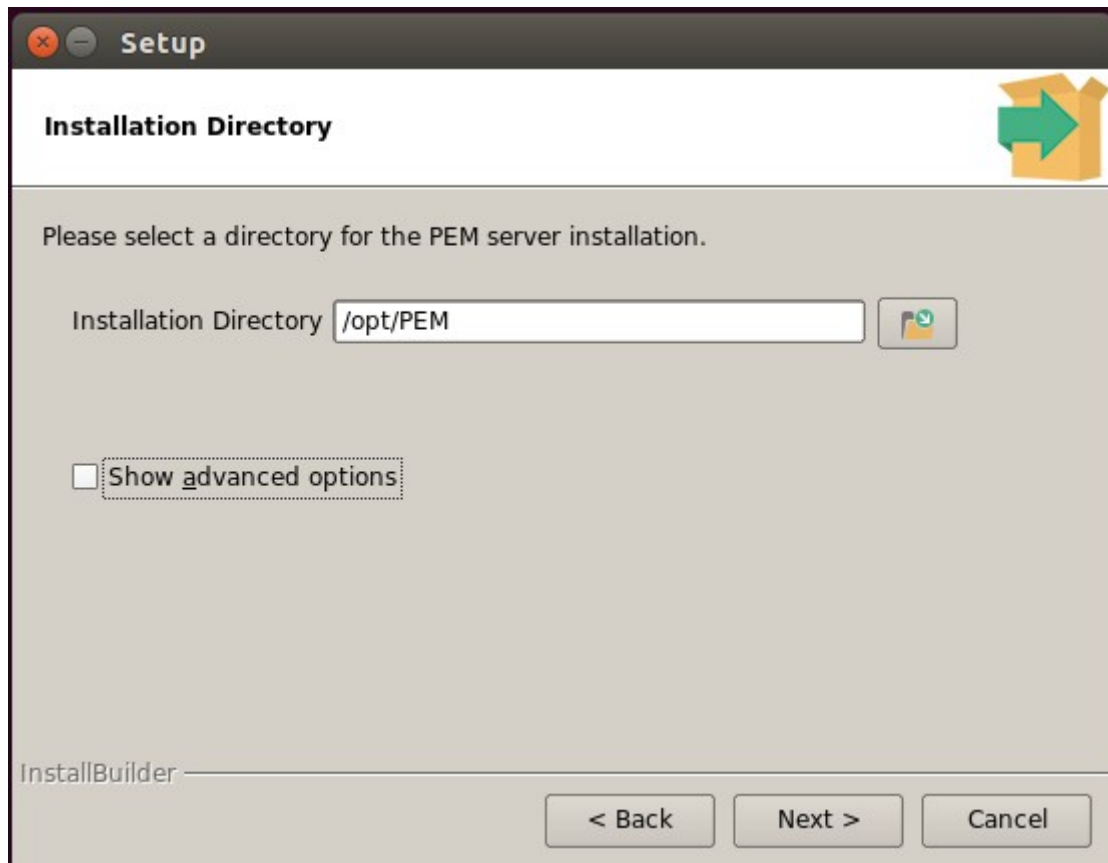
Email address

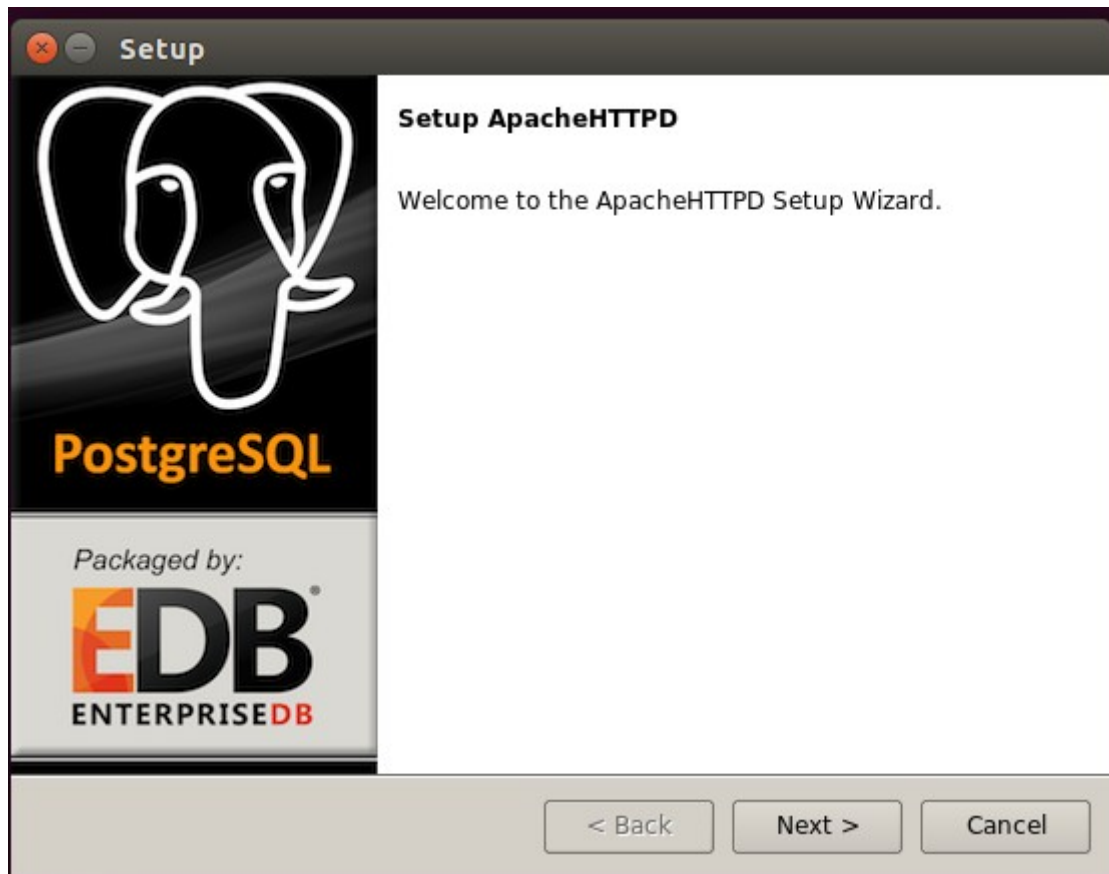
Password

InstallBuilder

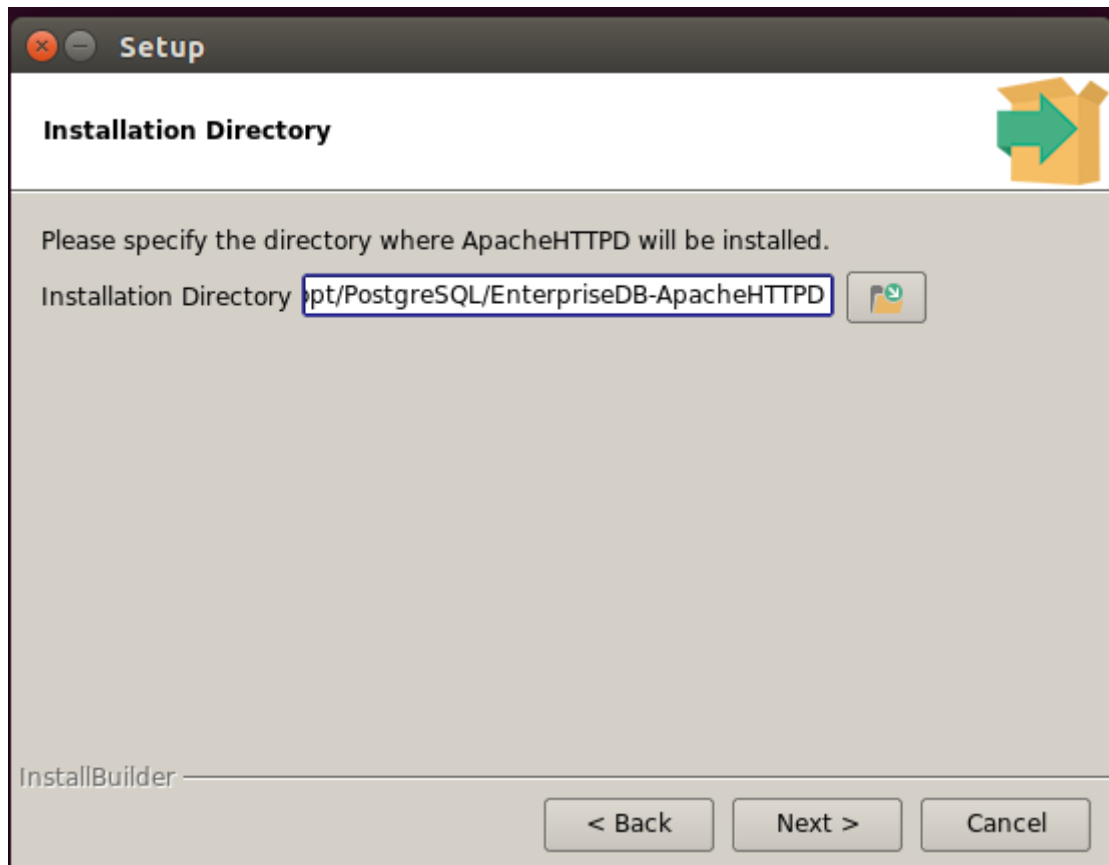
< Back Next > Cancel

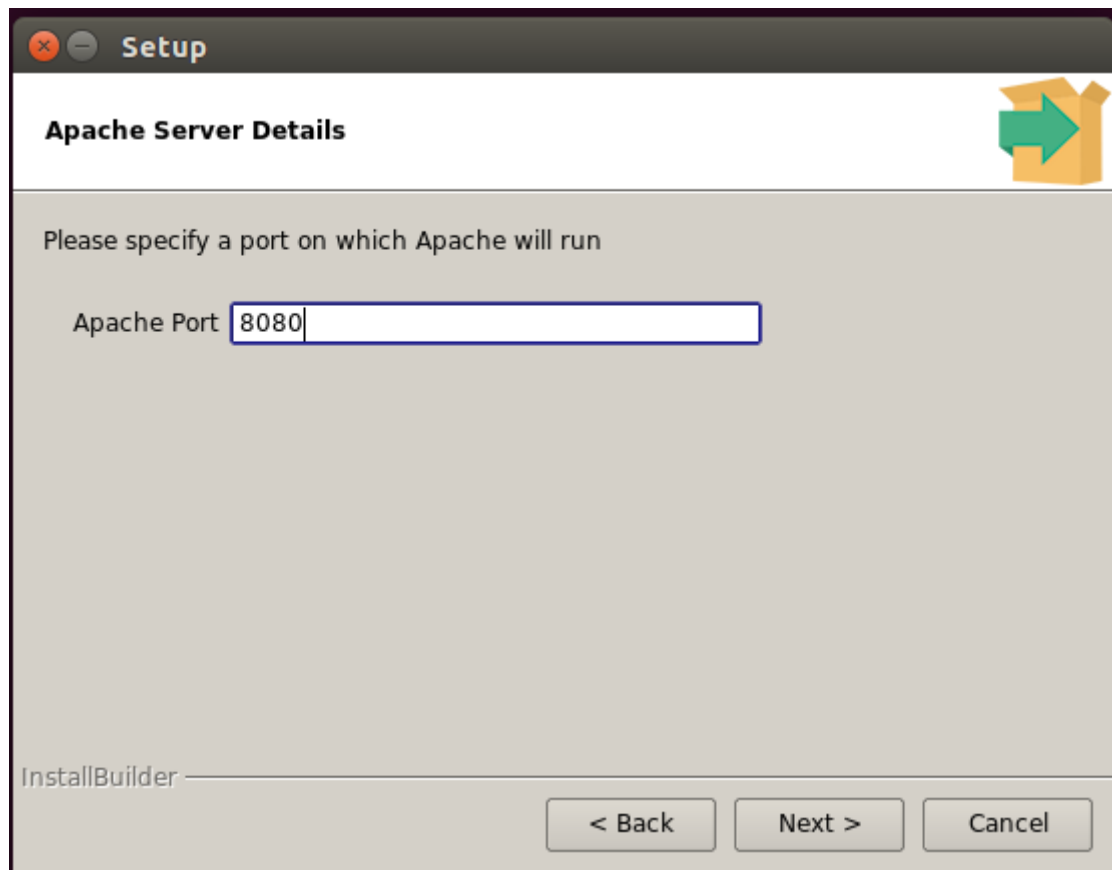
Practicas Avanzado PostgreSQL

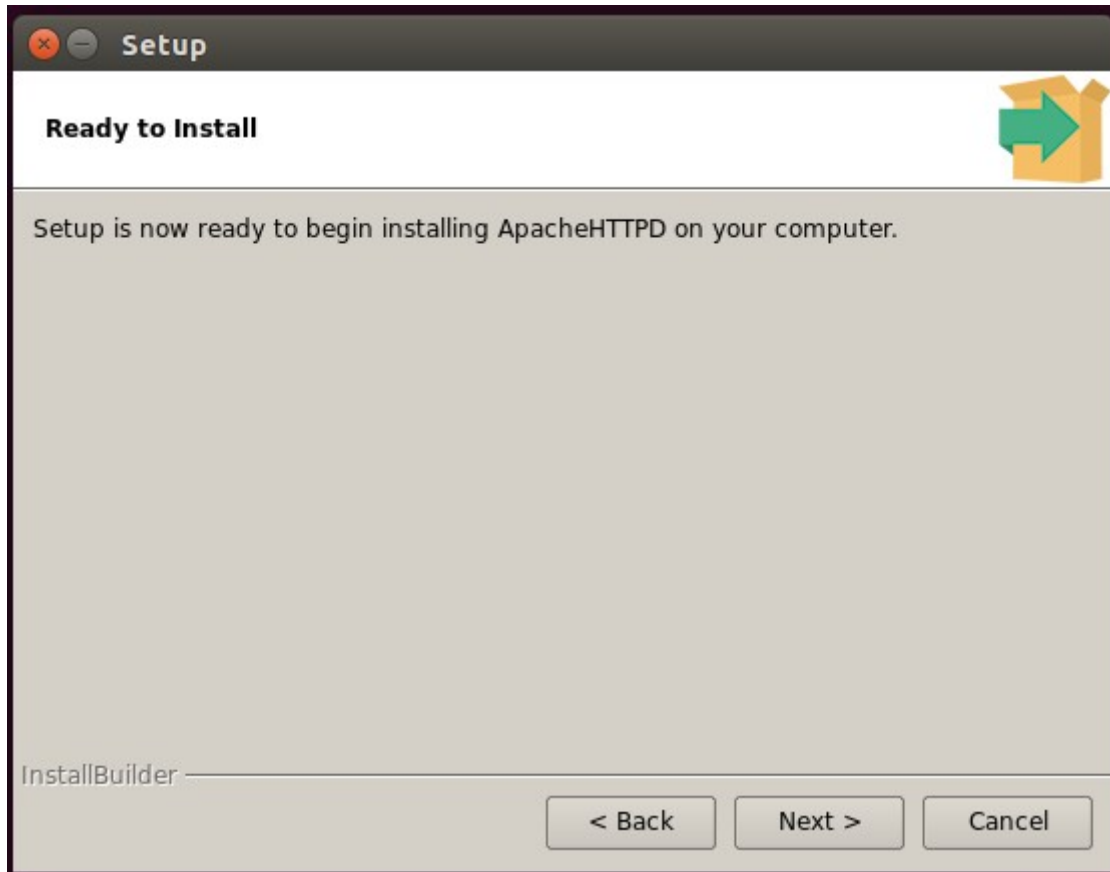




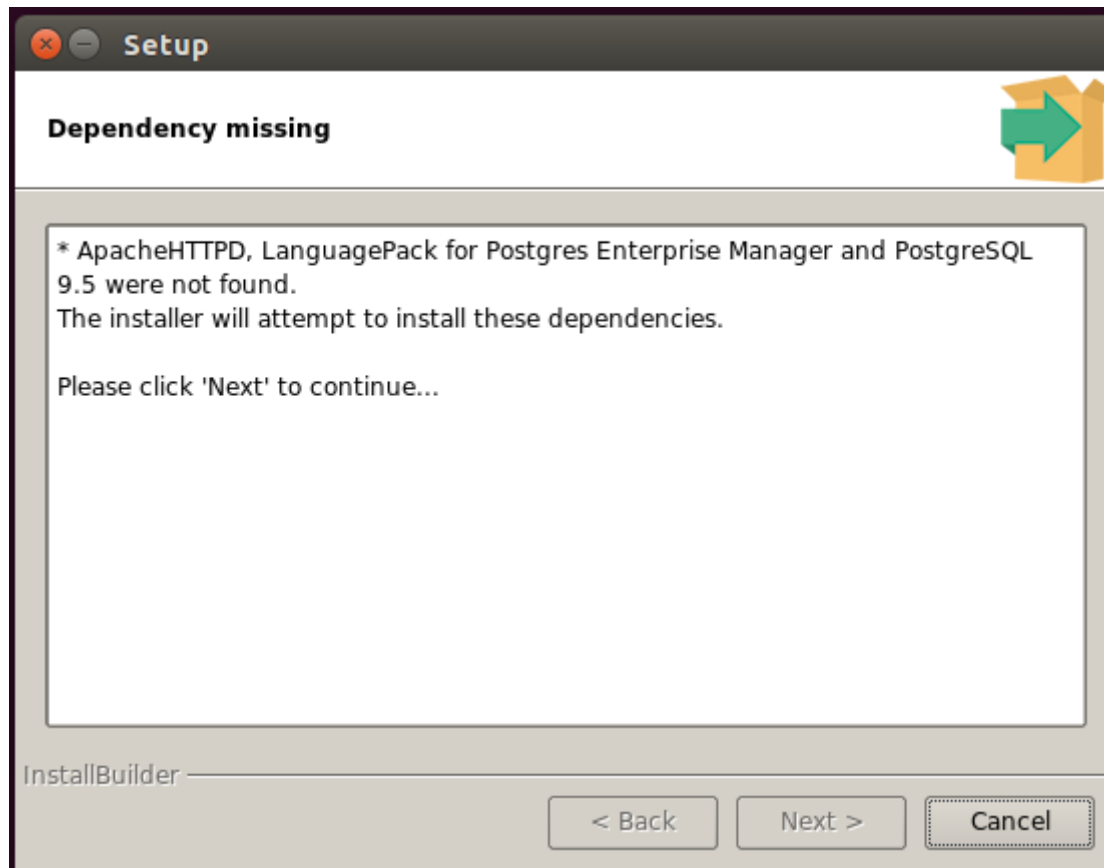
Practicas Avanzado PostgreSQL











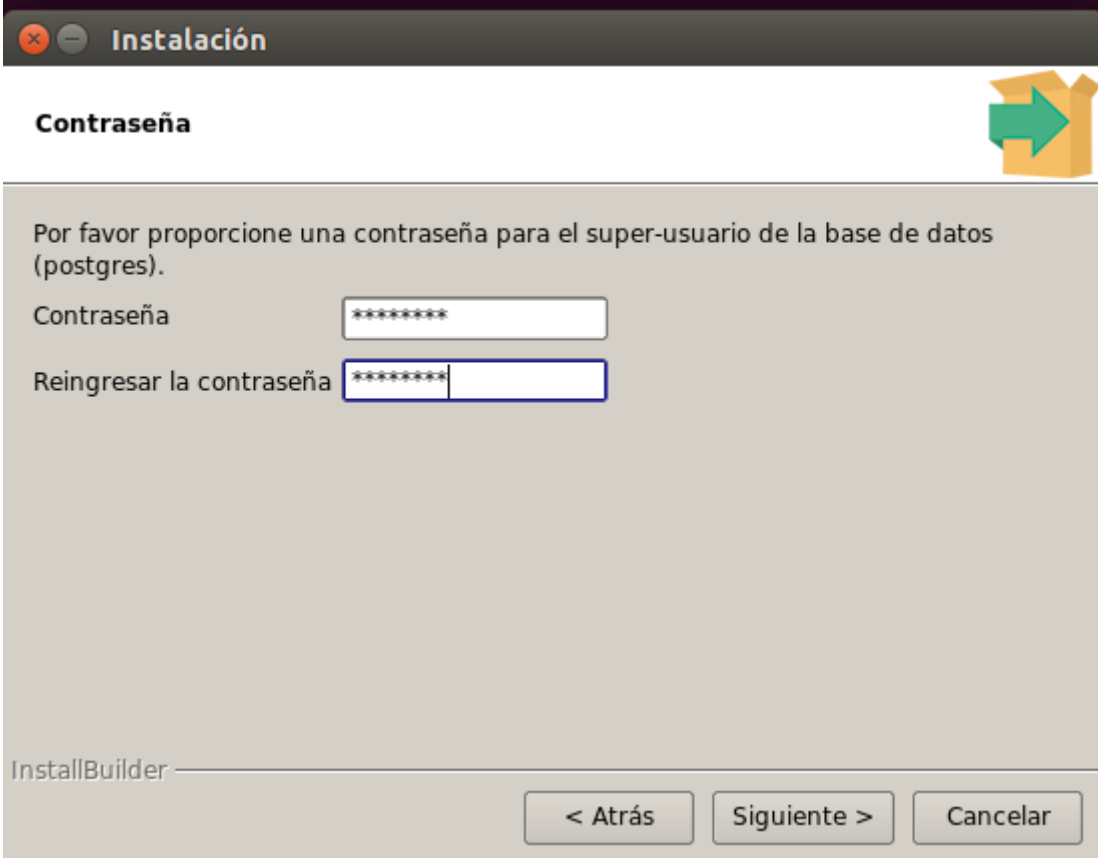
Practicas Avanzado PostgreSQL



Practicas Avanzado PostgreSQL



Practicas Avanzado PostgreSQL



Instalación

Contraseña

Por favor proporcione una contraseña para el super-usuario de la base de datos (postgres).


Contraseña

Reingresar la contraseña

InstallBuilder

< Atrás Siguiete > Cancelar

Practicas Avanzado PostgreSQL



Instalación

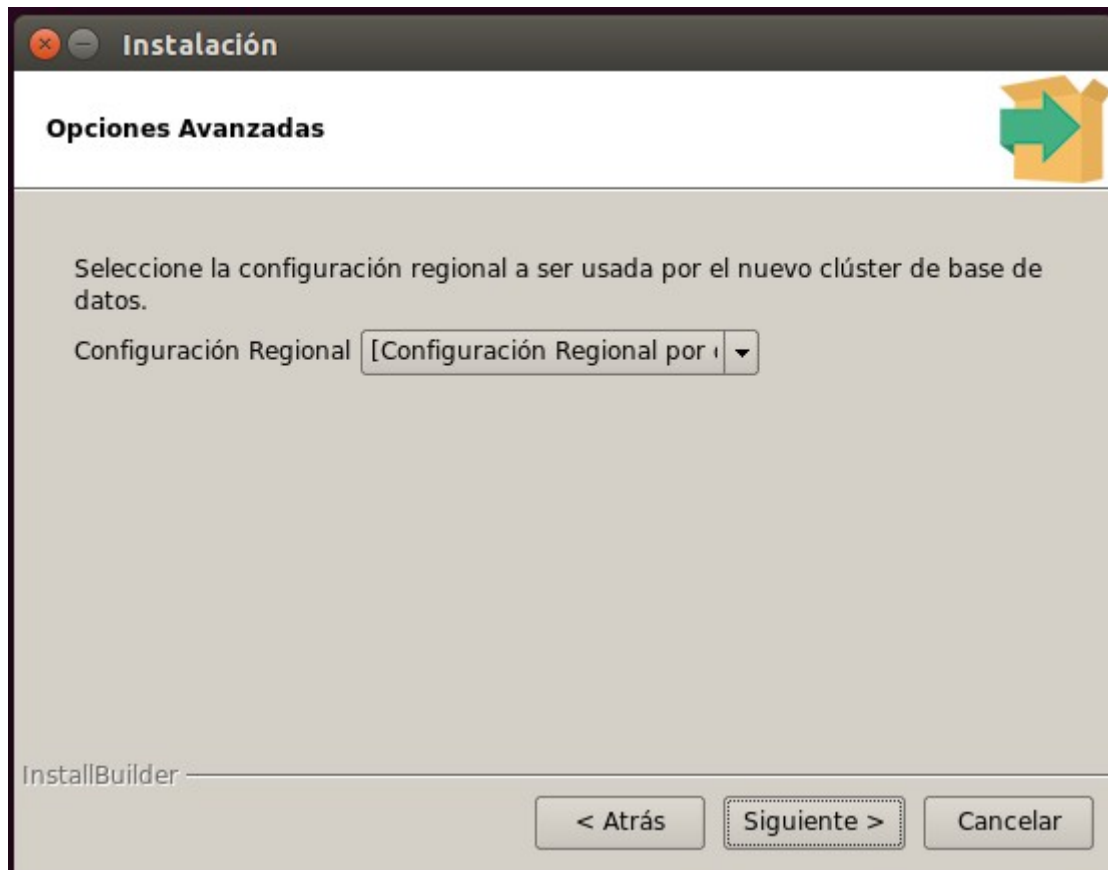
Puerto

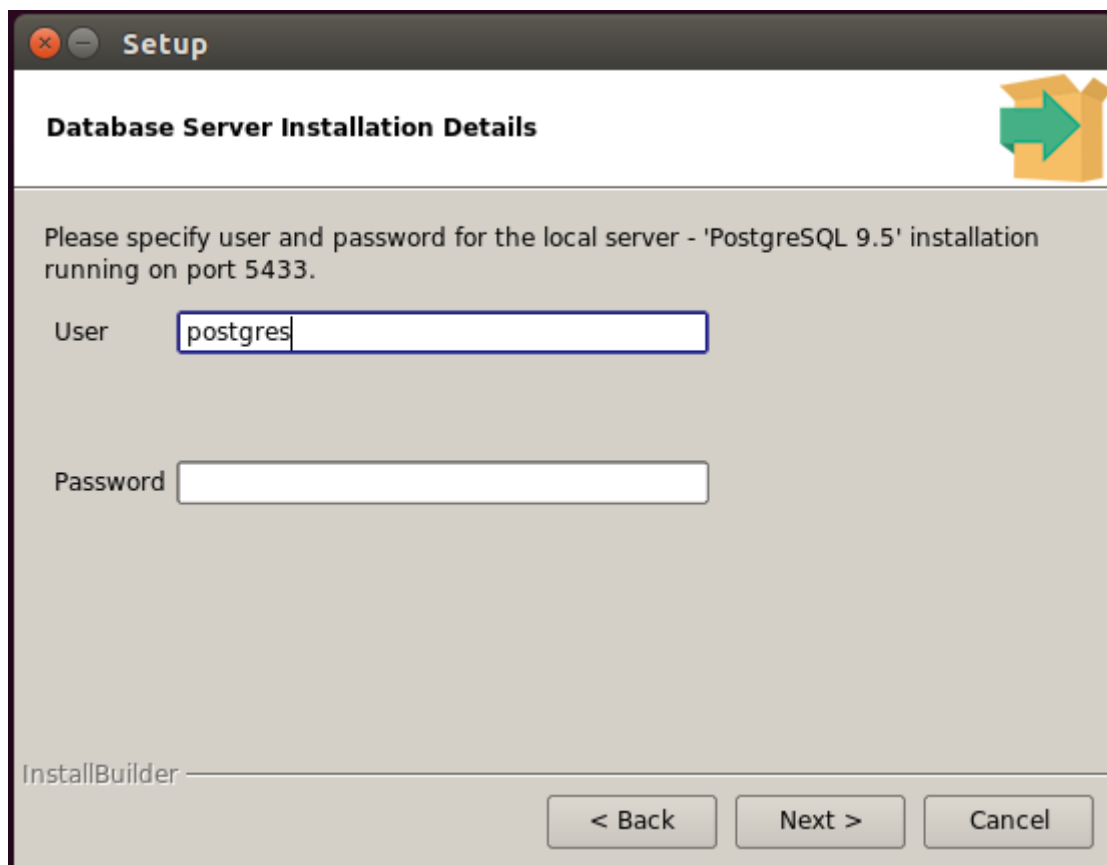
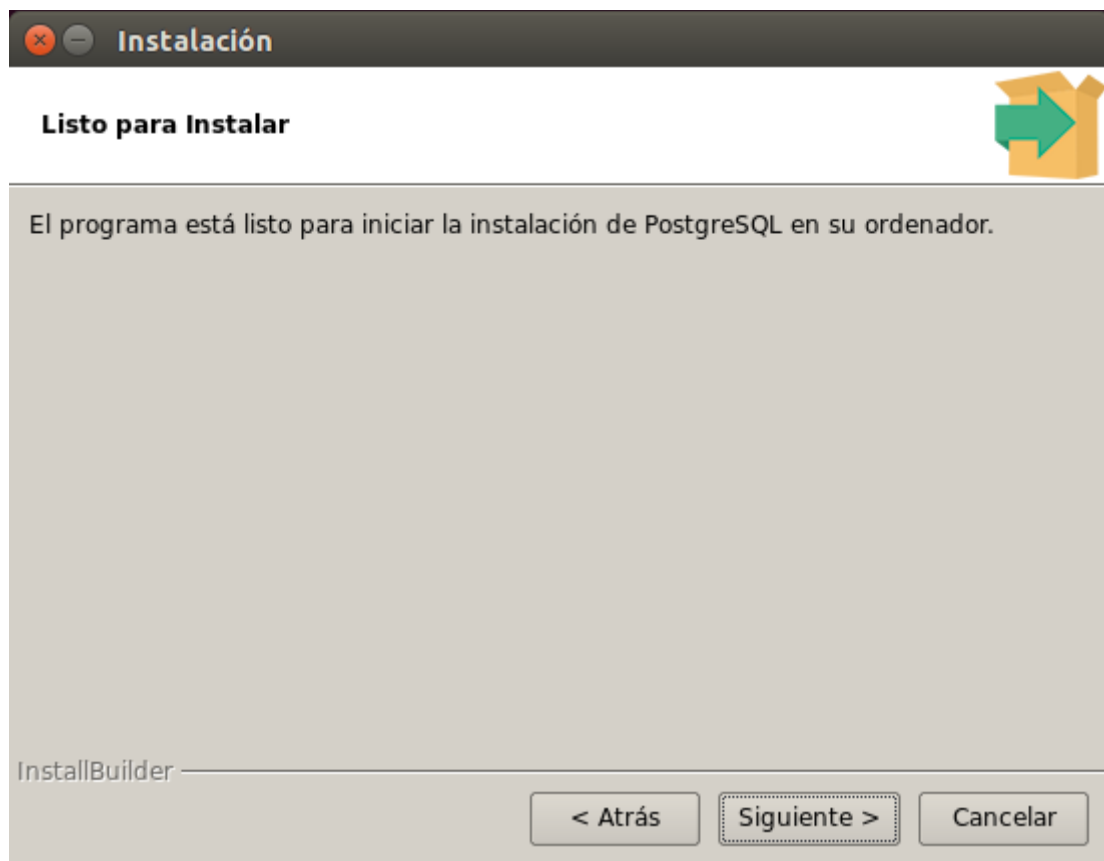
Por favor seleccione un número de puerto en el que el servidor debería escuchar.

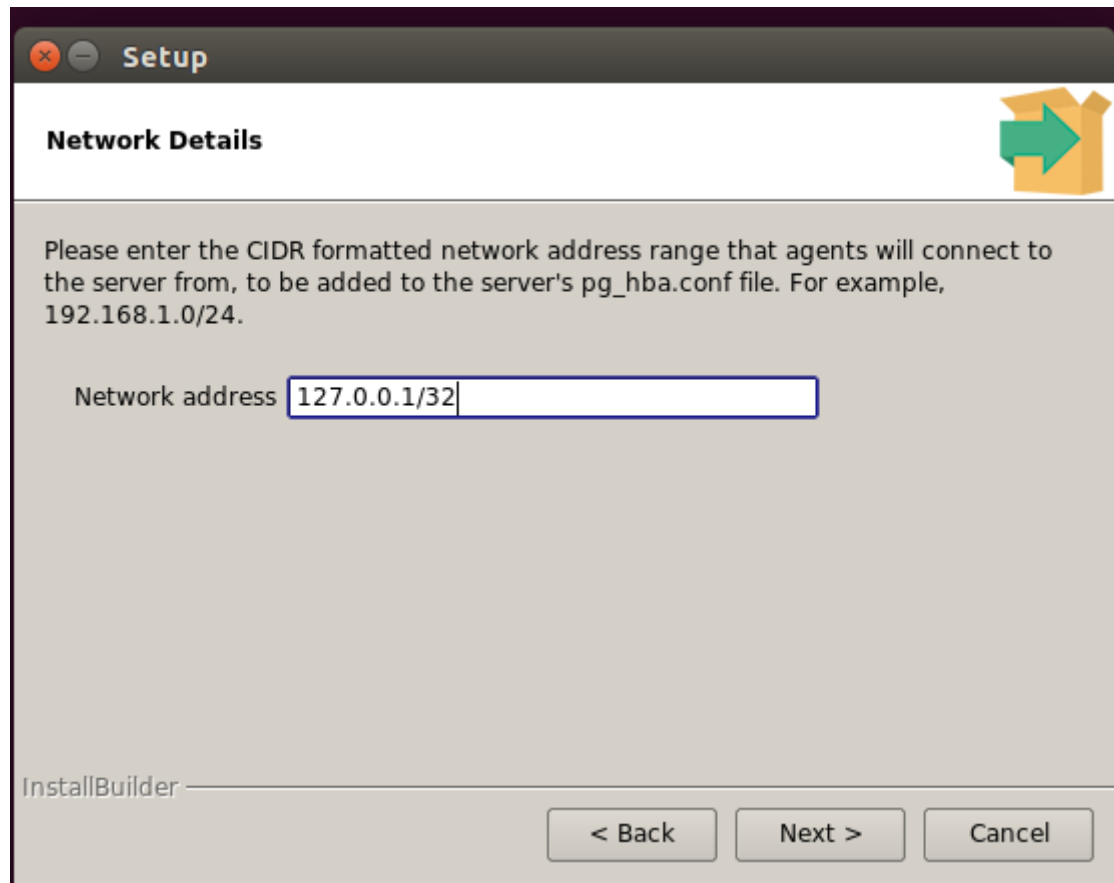
Puerto

InstallBuilder

< Atrás Siguiete > Cancelar







The image shows a 'Setup' window titled 'Network Details'. It contains a text box for 'Network address' with the value '127.0.0.1/32'. The window has a title bar with standard OS controls and a 'Setup' title. A green arrow icon is in the top right corner. The bottom of the window features three buttons: '< Back', 'Next >', and 'Cancel'. The text 'InstallBuilder' is visible in the bottom left corner of the window's content area.

Setup

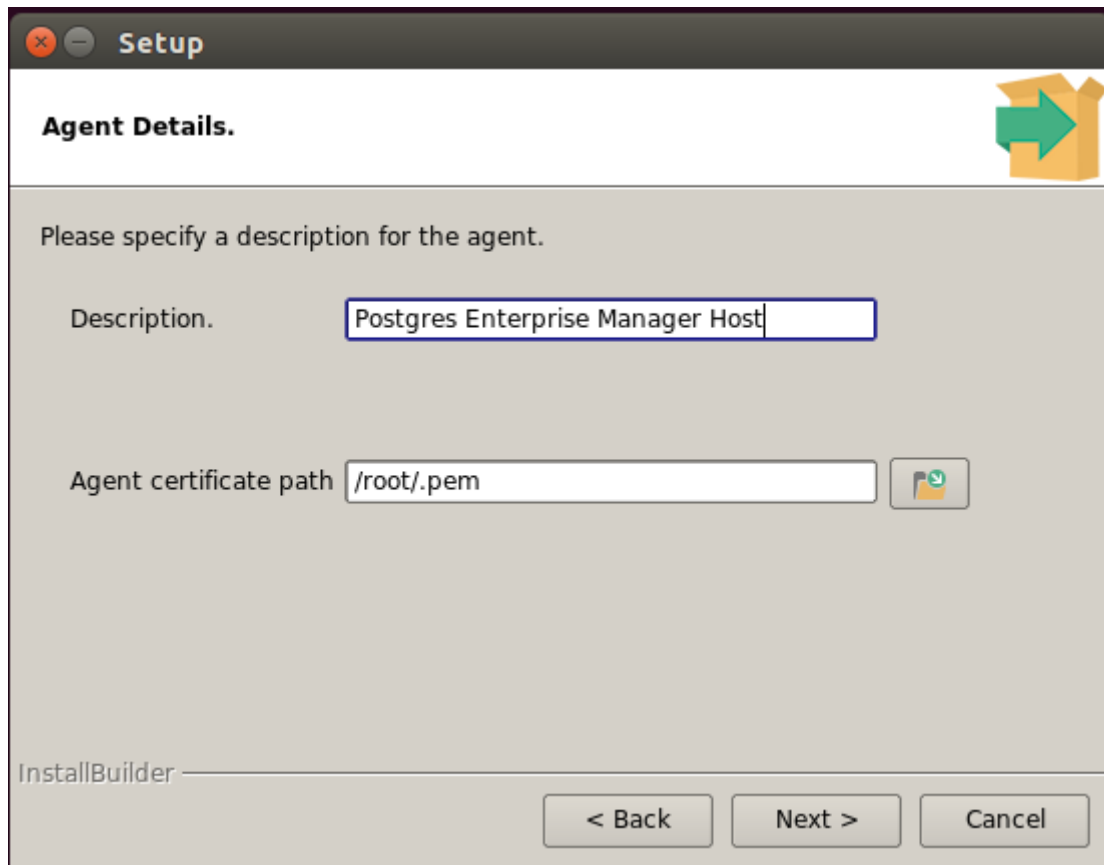
Network Details

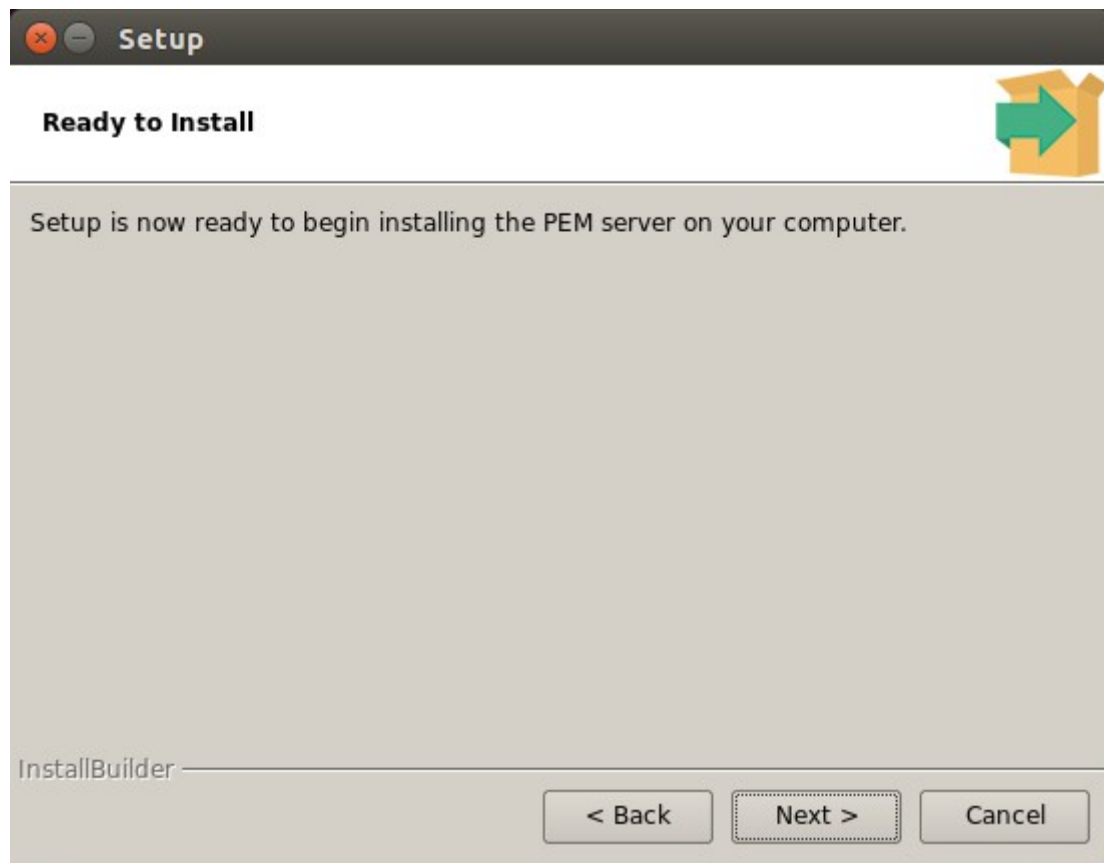
Please enter the CIDR formatted network address range that agents will connect to the server from, to be added to the server's pg_hba.conf file. For example, 192.168.1.0/24.

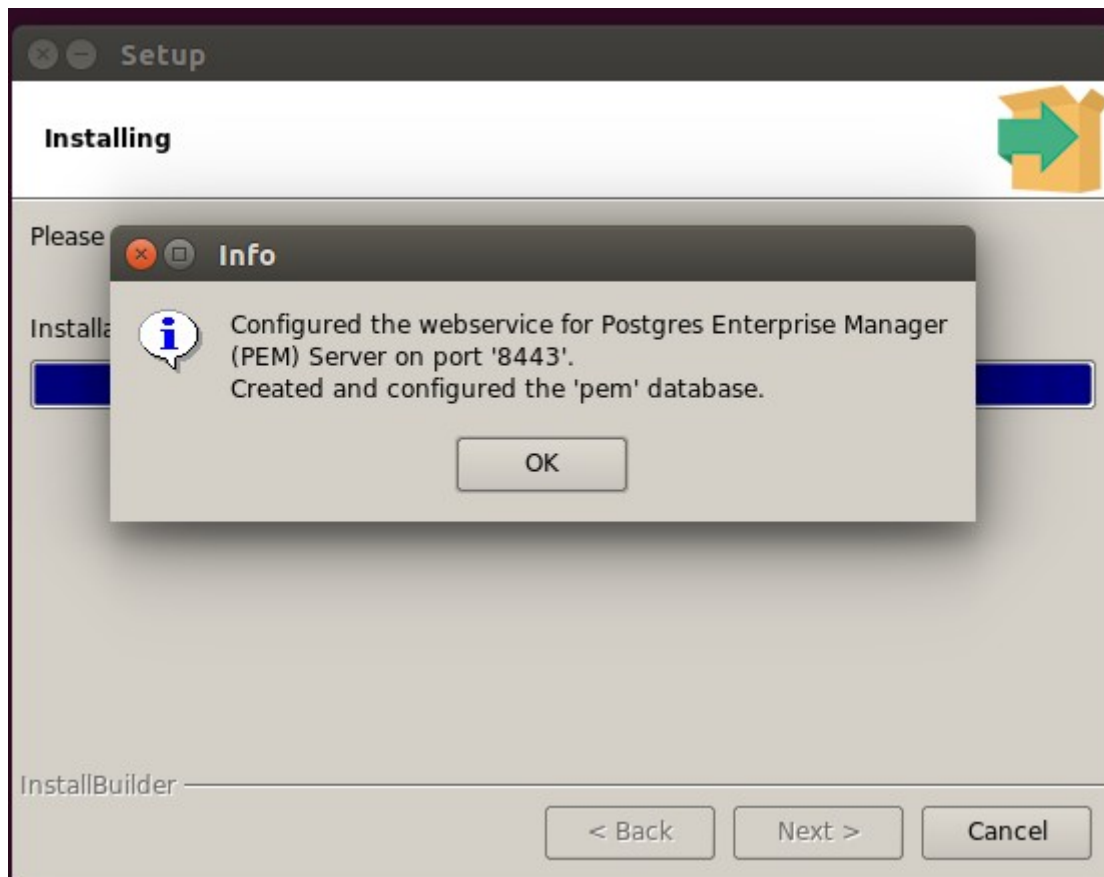
Network address

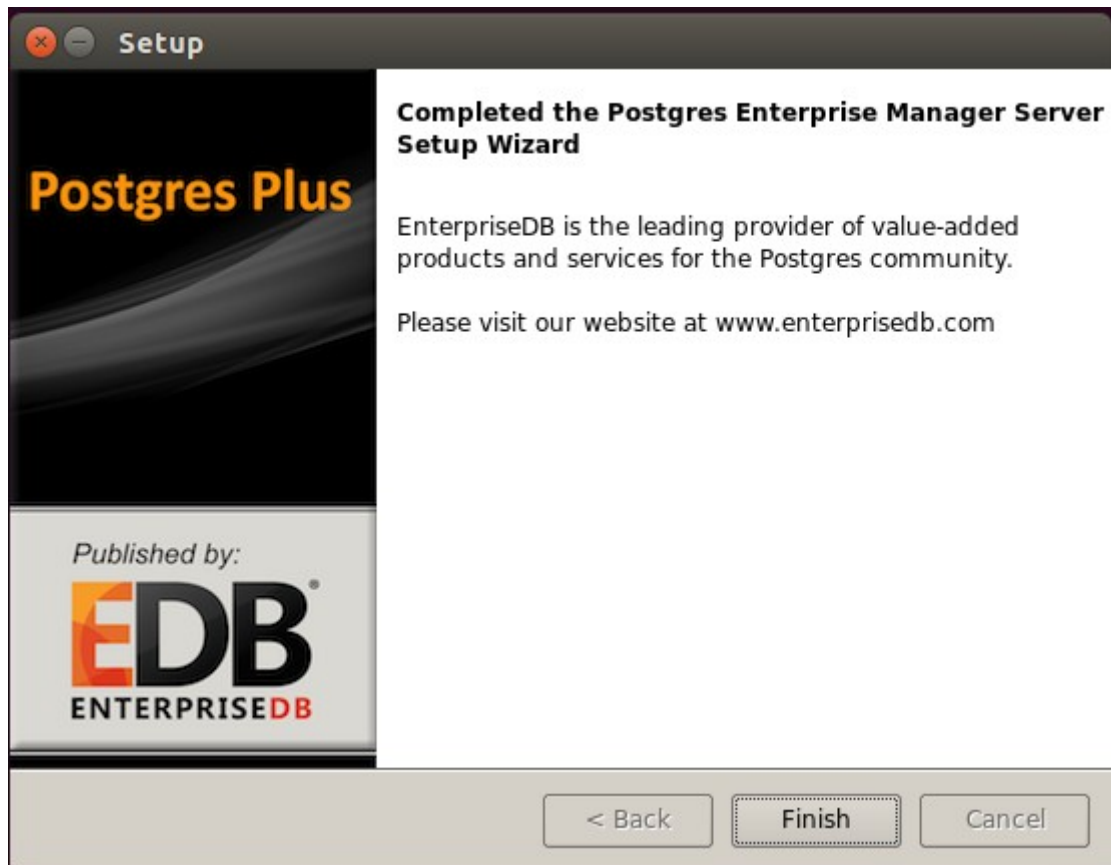
InstallBuilder

< Back Next > Cancel



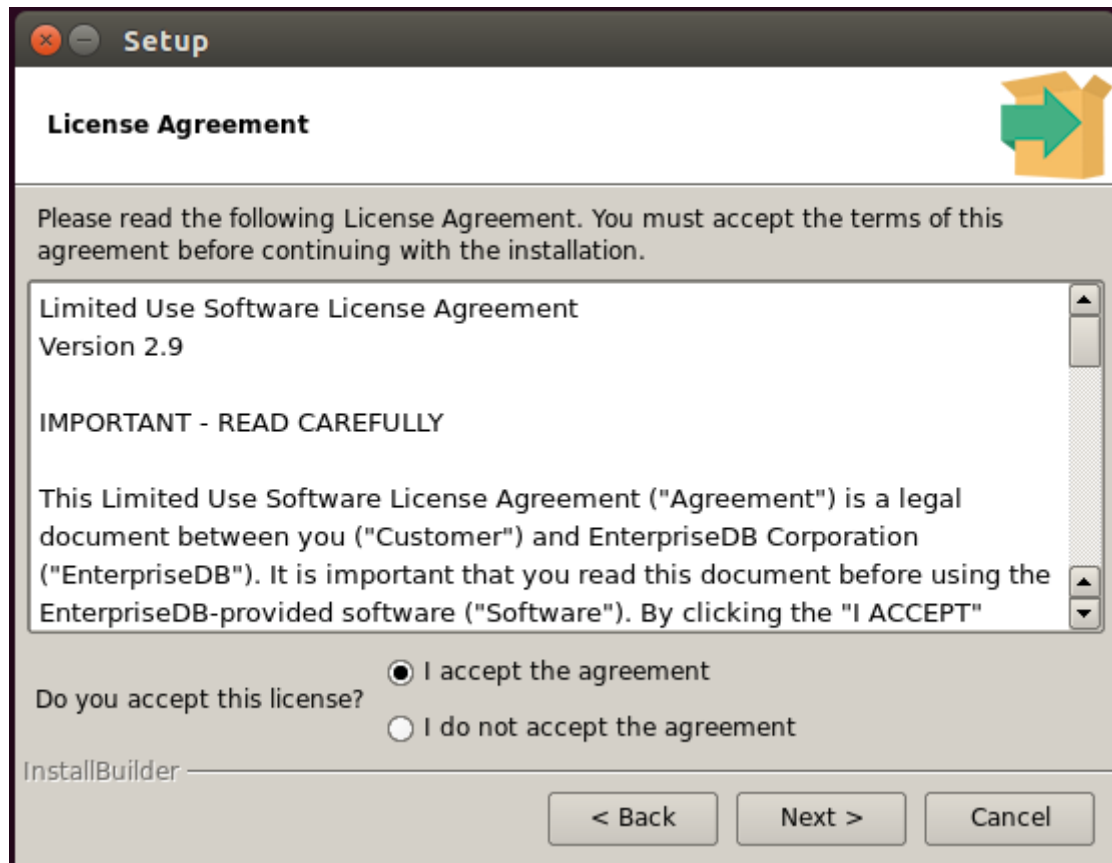


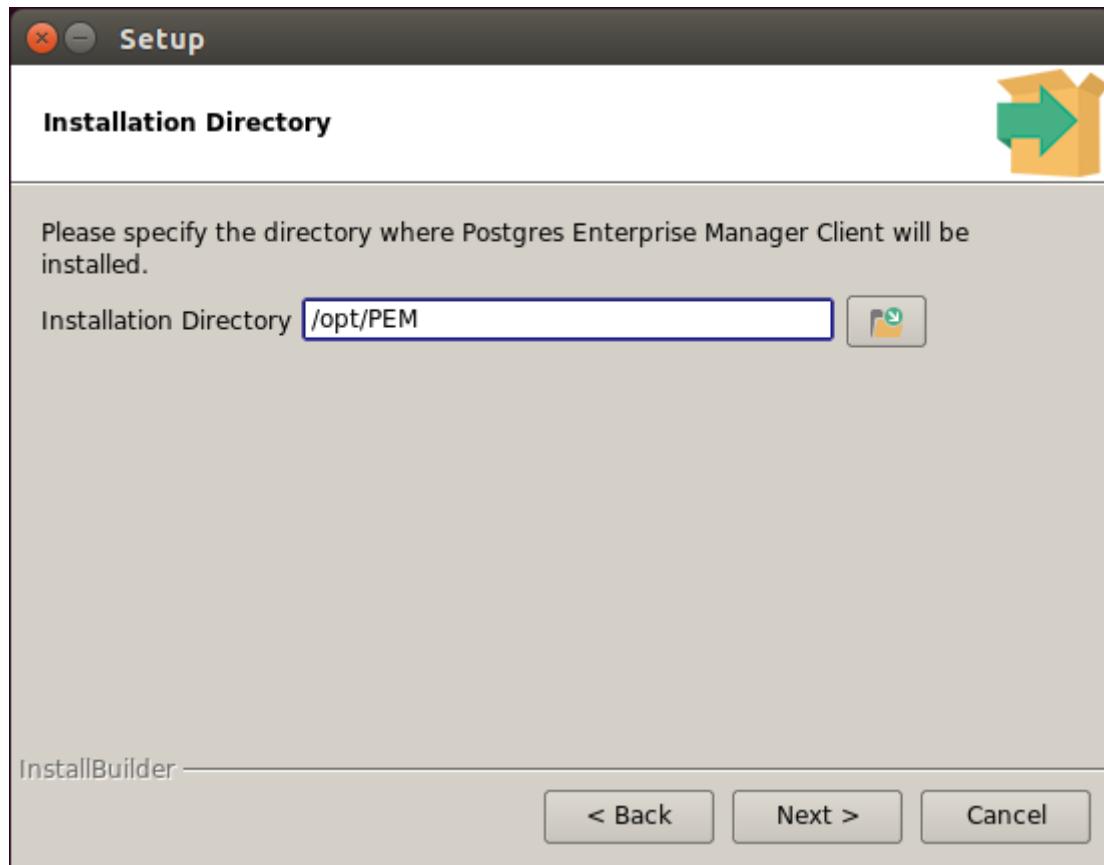


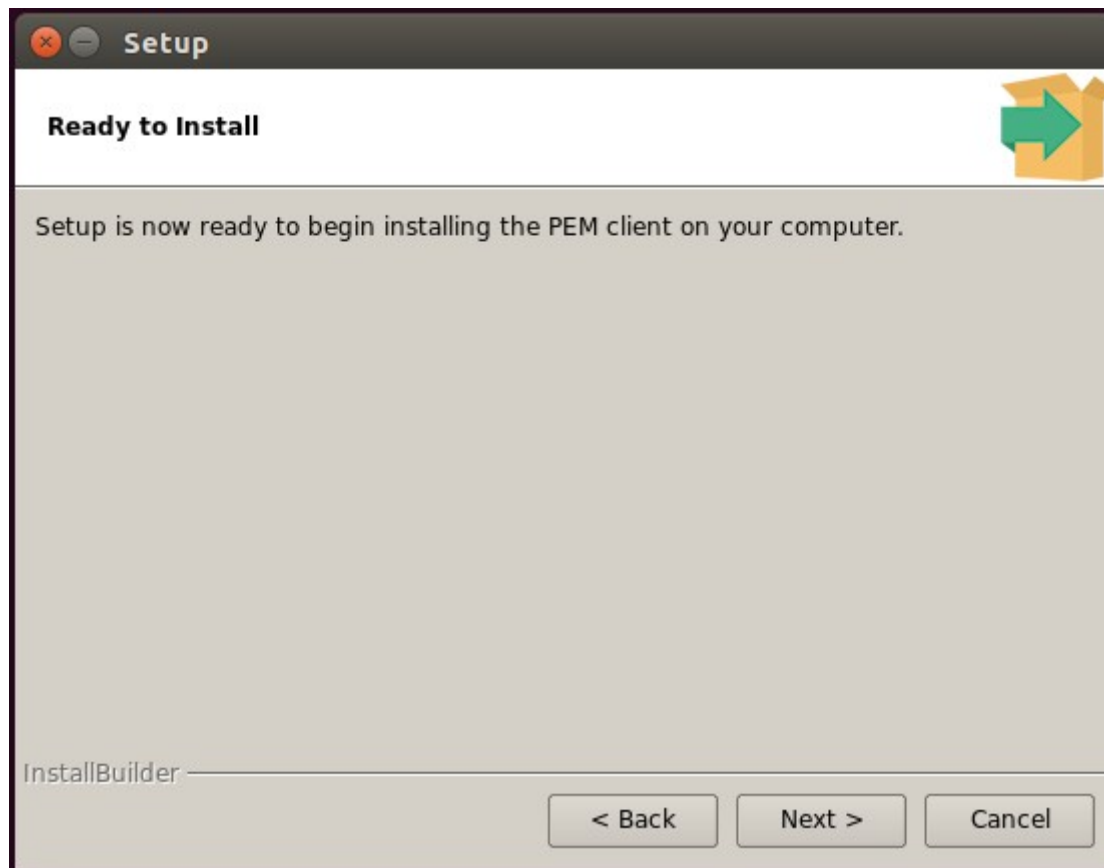


- **Descargar e instalar Postgres Enterprise Manager Client**



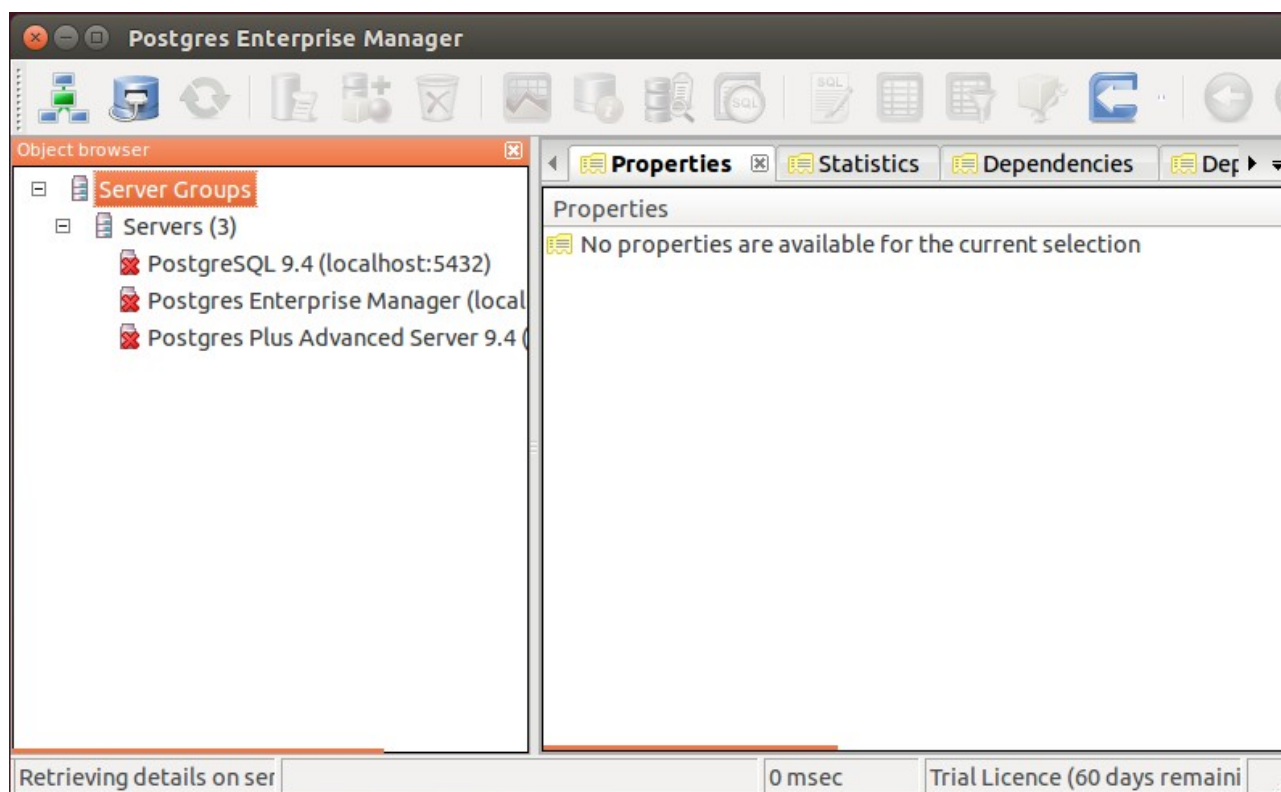






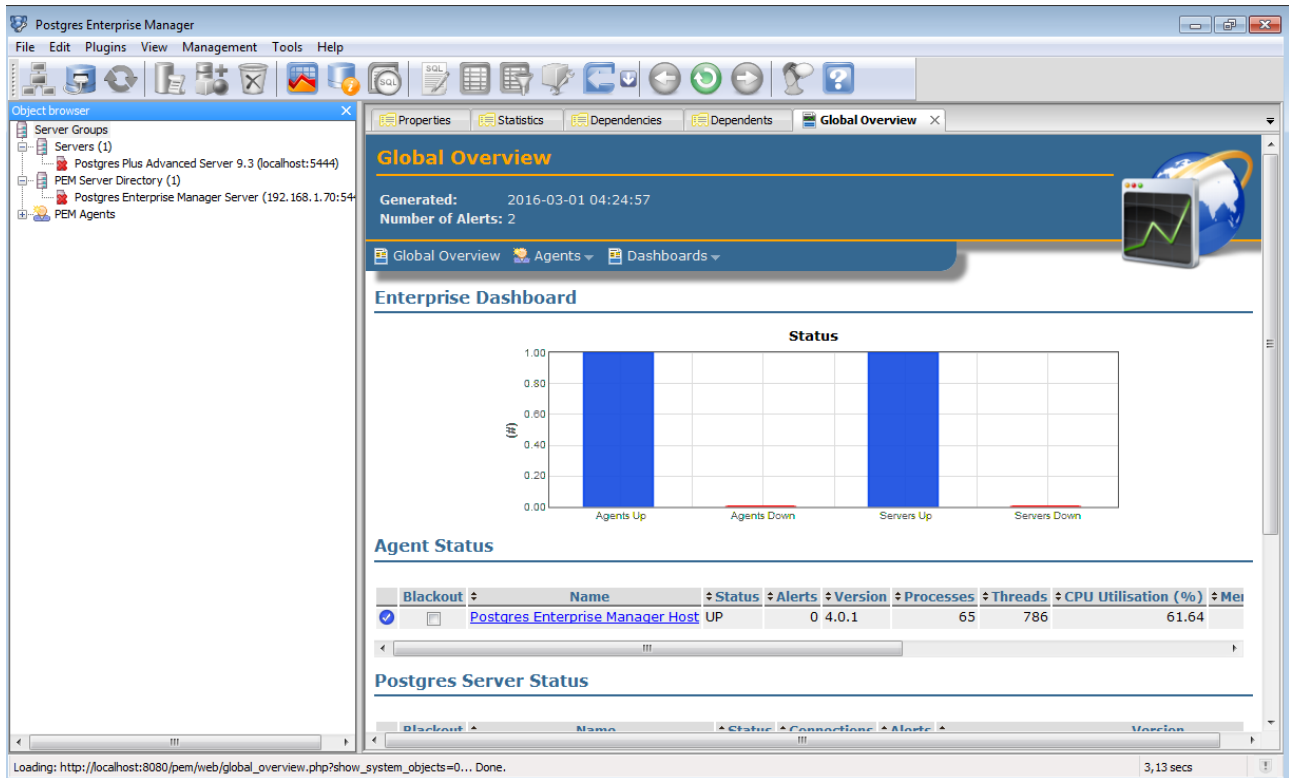


- Abrir PEM Client

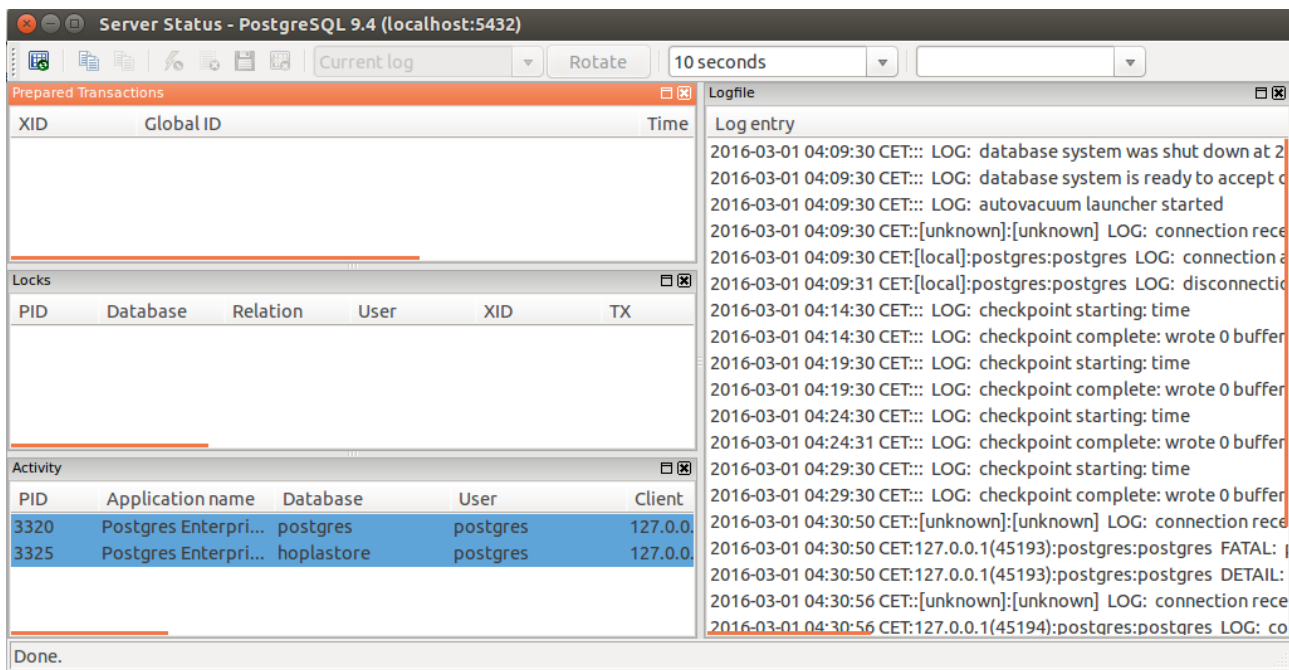


Practicar Avanzado PostgreSQL

- El Cluster predeterminado enlazado se ejecuta en el puerto 5432 con PEM agent para coleccionar la monitorización de datos
- Mirar panel de control global para PEM server

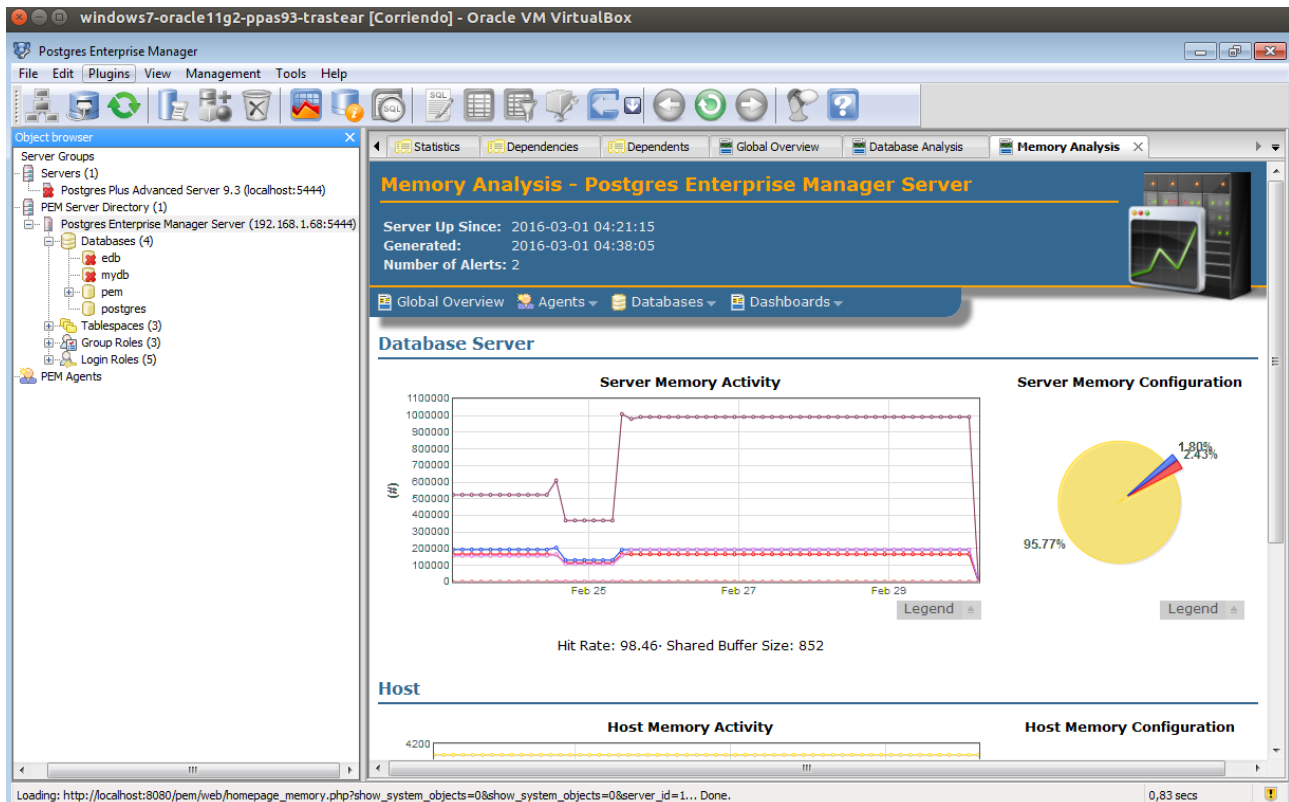


- Mirar la actividad actual de la base de datos del panel de control para su cluster predeterminado



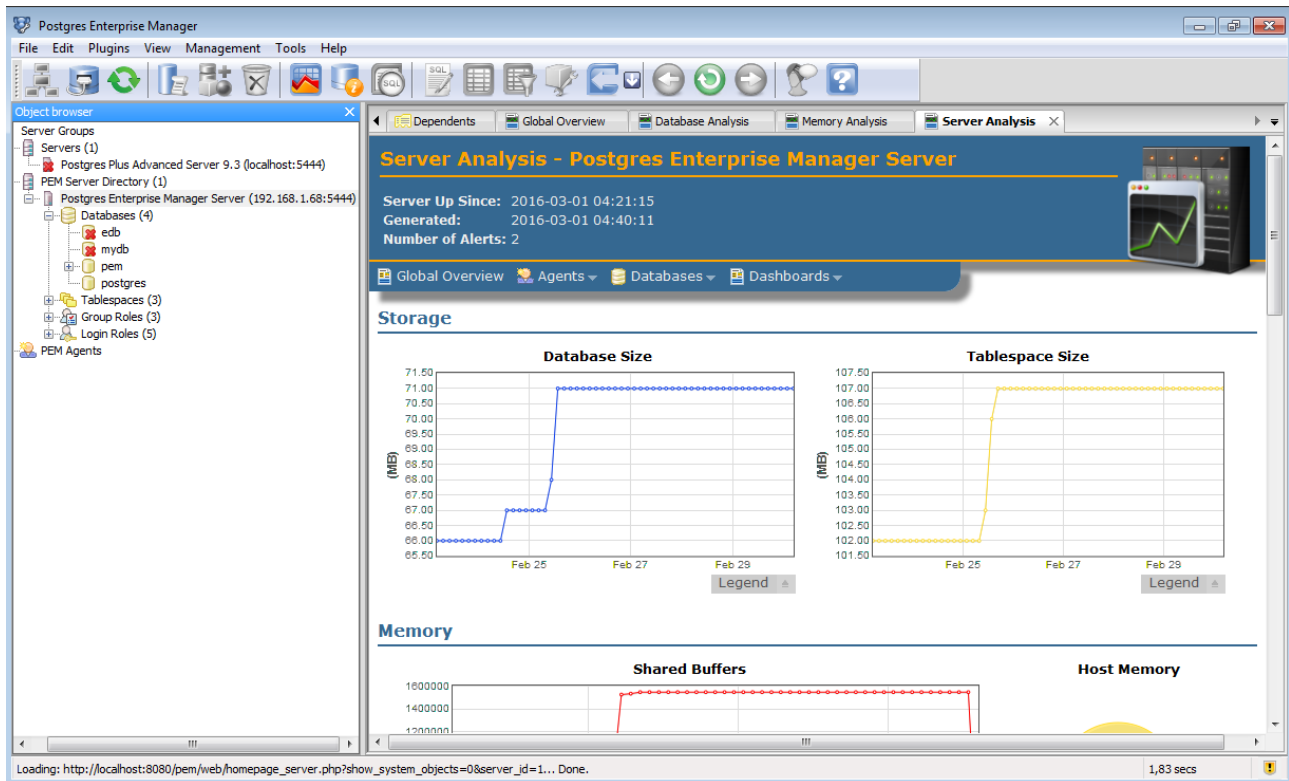
Practicas Avanzado PostgreSQL

- Ver el panel de control de la memoria para su cluster predeterminado



- Mirar información OS para el servidor donde el cluster predeterminado se esté ejecutando

Practicas Avanzado PostgreSQL



2.8.3 Ejercicio 2

- Escriba una consulta para ver la sesión actual que está ejecutándose en tu cluster predeterminado.
- Escriba una consulta para ver cuantos usuarios están en estado idle
- Abra otro terminal y conéctelo con la base de datos hoplastore utilizando `psql`
- Encuentra la id del proceso para los usuarios recientemente conectados utilizando la terminal previa `psql`
- Escriba un comando para deshacer la ejecución de la consulta actual
- Escriba un comando para terminar la sesión `psql` recién creada

2.8.4 Solución 2

- **Escriba una consulta para ver la sesión actual que está ejecutándose en tu cluster predeterminado.**

```
hoplastore=> select * from pg_stat_activity;
```

datid	datname	pid	usesysid	username	application_name	client_addr
client_hostname	client_port	backend_start	xact_start	query_start	state_change	waiting
rt	state	backend_xid	backend_xmin	query		

Practicar Avanzado PostgreSQL

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
17158 | hoplastore | 3506 | 16393 | hoplastore | psql.bin | | |
-1 | 2016-03-01 04:51:09.904907+01 | 2016-03-01 04:51:11.635163+01 | 2016-03-
01 04:51:1
1.635163+01 | 2016-03-01 04:51:11.635169+01 | f | active | | 1567 |
select * from pg_stat_activity;
(1 row)
```

- **Escriba una consulta para ver cuantos usuarios están en estado idle**

```
hoplastore=> select * from pg_stat_activity where state like '%idle in transaction%';
datid | datname | pid | usesysid | username | application_name | client_addr |
client_hostname | client_port | backend_start | xact_start | query_start | state_change
| waiting | state | backend_xid |
backend_xmin | query
-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+
(0 rows)
```

- **Abra otro terminal y conéctelo con la base de datos hoplastore utilizando psql**
- **Encuentra la id del proceso para los usuarios recientemente conectados utilizando la terminal previa psql**

```
hoplastore=> select * from pg_stat_activity;
datid | datname | pid | usesysid | username | application_name | client_addr |
client_hostname | client_port | backend_start | xact_start |
query_start |
state_change | waiting | state | backend_xid | backend_xmin |
query
-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+
17158 | hoplastore | 3605 | 16393 | hoplastore | psql.bin | | |
-1 | 2016-03-01 04:57:23.338485+01 | | |
| 2016-03-01 04:57:23.340179+01 | f | idle | | |
17158 | hoplastore | 3506 | 16393 | hoplastore | psql.bin | | |
-1 | 2016-03-01 04:51:09.904907+01 | 2016-03-01 04:57:37.98388+01 | 2016-03-01
04:57:3
7.98388+01 | 2016-03-01 04:57:37.983883+01 | f | active | | 1567 |
select * from pg_stat_activity;
(2 rows)
```

- **Escriba un comando para deshacer la ejecución de la consulta actual**

```
hoplastore=> select pg_cancel_backend(3605);
pg_cancel_backend
-----
t
(1 row)
```

- **Escriba un comando para terminar la sesión psql recién creada**

```
hoplastore=> select pg_terminate_backend(3605);
pg_terminate_backend
-----
t
(1 row)
```

2.9 Módulo 8

2.9.1 Ejercicio 1

- Crear una función plpgsql
- Declarar dos variables en la sección declarar
- Asignar el valor 1 en la primera variable
- Asignar el valor 2 en la segunda variable
- Mostrar la diferencia entre numéricos en la pantalla utilizando mensajes de noticia

2.9.2 Solución 1

- Crear una función plpgsql
- Declarar dos variables en la sección declarar
- Asignar el valor 1 en la primera variable
- Asignar el valor 2 en la segunda variable
- Mostrar la diferencia entre numéricos en la pantalla utilizando mensajes de noticia

```
# Creación de la función
create or replace function resta_numeros() returns numeric as
$$
declare
dos numeric;
uno numeric;
resultado numeric;
begin
dos := 2;
uno := 1;
resultado := dos-uno;
raise notice 'Resultado %',resultado;
```

```
return 0;
end;
$$ language plpgsql;

# Ejecución
-- Executing query:

select resta_numeros();
NOTICE: Resultado 1

Total query runtime: 12 ms.
1 row retrieved.
```

2.10 Módulo 9

2.10.1 Ejercicio 1

- El equipo de base de datos quiere examinar que está pasando en el cache del buffer compartido a tiempo real. Podrás encontrar una extensión interesante de PostgreSQL llamada "pg_buffercache". Realizar la instalación y prueba la funcionalidad de este módulo de extensión.

2.10.2 Solución 1

- **El equipo de base de datos quiere examinar que está pasando en el cache del buffer compartido a tiempo real. Podrás encontrar una extensión interesante de PostgreSQL llamada "pg_buffercache". Realizar la instalación y prueba la funcionalidad de este módulo de extensión.**

```
# Conectarse a la base de datos donde se empleará la extensión con un superusuario
postgres=> \c hoplastore postgres
You are now connected to database "hoplastore" as user "postgres".

# Crear la extensión
hoplastore=# create extension pg_buffercache ;
CREATE EXTENSION

# Reiniciar la instancia para vaciar los buffers
hoplastore=# SELECT c.relname, count(*) AS buffers
hoplastore-# FROM pg_buffercache b INNER JOIN pg_class c
hoplastore-# ON b.relfilenode = pg_relation_filenode(c.oid) AND
hoplastore-# b.reldatabase IN (0, (SELECT oid FROM pg_database
hoplastore(=# WHERE datname = current_database()))
hoplastore-# GROUP BY c.relname
hoplastore-# ORDER BY 2 DESC
hoplastore-# LIMIT 10;
```


relname	buffers
pg_attribute	21
pg_class	9
pg_proc	8
pg_proc_oid_index	6
pg_attribute_relid_attnum_index	6
pg_class_oid_index	5
pg_type	4
pg_proc_prname_args_nsp_index	4
pg_opclass	3
pg_statistic_relid_att_inh_index	3

(10 rows)

NOTA: Se ve que ninguna de las tablas de usuario están entre los bloques calientes de la base de datos. Para ello vamos a cargar otra extensión para subir a memoria una tabla en concreto.

```
# Crear la extensión pg_prewarm
hoplastore=# create extension pg_prewarm ;
CREATE EXTENSION
```

```
# Subimos la tabla customers a memoria
hoplastore=# select pg_prewarm ('hoplastore.customers');
pg_prewarm
```

```
-----
      488
(1 row)
```

```
# Se vuelve a ejecutar el buffercache
hoplastore=# SELECT c.relname, count(*) AS buffers
FROM pg_buffercache b INNER JOIN pg_class c
ON b.relfilenode = pg_relation_filenode(c.oid) AND
b.reldatabase IN (0, (SELECT oid FROM pg_database
WHERE datname = current_database()))
GROUP BY c.relname
ORDER BY 2 DESC
LIMIT 10;
```

relname	buffers
customers	488
pg_attribute	23
pg_proc	19
pg_class	9
pg_proc_oid_index	7
pg_attribute_relid_attnum_index	7
pg_proc_prname_args_nsp_index	6
pg_description	5
pg_class_oid_index	5
pg_depend_reference_index	5

(10 rows)

