1. Instalação do VM:

* Baixar a ISO do Ubuntu Server, que contêm o Ubuntu Terminal;
* Durante a instalação da VM, configurar a placa de rede “Adaptador 1” utilize a configuração “Conectado a:” : “Placa em Modo Bridge”; isto fará a geração de um endereço IP para a VM;
* Durante o processo de instalação do Ubuntu, quando chegarmos na fase de SELEÇÃO DE SOFTWARES deixe selecionado os itens: Standard System Utilities e OpenSSH Server (Para liberar acessos remotos).

2. Instalação do PostgreSQL:

* Acessar o link <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads> para baixar a versão do postgreSQL. Neste caso, será baixado o arquivo “run” e após o download altere o arquivo para um auto executável com comando chmod +x e execute-o com “.\postgre...run”.
* Se optar pelos comandos sudo, podemos utilizar da seguinte forma:

sudo apt-get update && sudo apt-get upgrade

sudo apt-get install postgresql-10 postgresql-contrib

sudo apt-get install pgadmin3

sudo -u postgres psql -c "ALTER USER postgres PASSWORD 'postgres';"

sudo -u postgres psql -c "CREATE DATABASE testdb;"

* Após a instalação altere as configurações do banco para permitir melhor controle de acessso. Abra o arquivo da seguinte forma:

sudo vim /etc/postgresql/9.5/main/pg\_hba.conf

* Altere as configurações da seguinte forma:

# 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

to

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all md5

# IPv4 local connections:

host all all 0.0.0.0/0 md5

# IPv6 local connections:

host all all ::1/128 md5

* Abra outro arquivo de configurações:

sudo vim /etc/postgresql/9.5/main/postgresql.conf

* Altere as seguintes configurações:

#listen\_addresses = 'localhost'

Para

listen\_addresses = '\*'

# Isto libera o banco para ser conectado de outras estações.

* Reinicialize o serviço:

sudo /etc/init.d/postgresql restart

ou

sudo service postgresql restart

* Verificar a situação do banco:

/etc/init.d/postgresql status

* Caminho rápido para acessar o banco de dados:

sudo -u postgres createuser --interactive --password [User Name]

sudo -u postgres [Database Name] testdb -O [User Name]

sudo service postgresql restart

sudo -u postgres psql postgres

postgres=# \password [User Name]

\q

sudo -su postgres

psql -U [User Name] -d [Database Name] -W

* Acesse o banco da seguinte forma, utilizando a senha “postgres”:

sudo -u postgres psql postgres

ou

sudo -su postgres

psql [Database Name] [User Name]

Exemplo: psql testdb postgres

ou

psql -U [User Name] -d [Database Name] -W

Exemplo: psql -U marco -d desenv -W

* Crie um usuário:

CREATE USER ubuntu\_user [SUPERUSER];

ou

# No prompt do Linux:

sudo -u postgres createuser --interactive --password marco

* Definição da senha do usuário:

ALTER USER ubuntu\_user WITH PASSWORD 'password';

ou

#No prompt do linux:

sudo -u postgres createdb desenv -O user12

* Para criar um database com o usuário criado:

CREATE DATABASE mydatabase WITH OWNER ubuntu\_user;

* Dar privilégios ao usuário:

GRANT ALL PRIVILEGES ON mydatabase TO ubuntu\_user;

* Para sair do postgres:

\q

* Para listar os databases:

\l

* Para mudar de database:

\c [Database Name]

* Para mostrar os objetos do database:

\d

* Para mostrar os privilégios dos usuários:

\du

* Comandos básicos:

SELECT current\_user;

SELECT current\_database;

SELECT version();

SELECT \* FROM pg\_catalog.pg\_user;

* Autorização de privilégios diretamente na tabela:

sudo -su postgres

psql

CREATE USER geronimo PASSWORD ‘brasil2’

GRANT INSERT ON TABLE cliente TO geronimo;

\q

psql postgres geronimo

SELECT \* FROM cliente;

INSERT INTO cliente VALUES(1,’ASDFAS’);

* Autorização de privilégios para todas as tabelas:

sudo -su postgres

psql

CREATE USER geronimo WITH PASSWORD ‘brasil2’

CREATE TABLE produto( codigo INT, descricao VARCHAR(50));

GRANT INSERT ON ALL TABLES IN SCHEMA public TO geronimo;

\q

psql desenvolvimento geronimo

SELECT \* FROM produto;

INSERT INTO produto VALUES( 2, ‘PARAFUSO’);

Observação: Todos os databases possuem Schemas (agrupamento de objetos “Tables, Views, etc”), inicialmente um database quando criado já vem com o schema “public”, onde é armazenado inicialmente todos os objetos do database caso não for informado o schema ( CREATE TABLE **junho**.contrato( datBase DATE, contrato INT, … ).

* Listando todos os usuários do banco:

SELECT u.usename AS "Role name",

CASE

WHEN u.usesuper AND u.usecreatedb THEN

CAST('superuser, create database' AS pg\_catalog.text)

WHEN u.usesuper THEN

CAST('superuser' AS pg\_catalog.text)

WHEN u.usecreatedb THEN

CAST('create database' AS pg\_catalog.text)

ELSE

CAST('' AS pg\_catalog.text)

END AS "Attributes"

FROM pg\_catalog.pg\_user u

ORDER BY 1;

* Autorização para se conectar a um DATABASE:

GRANT CONNECT ON DATABASE database\_name TO username;

GRANT ALL PRIVILEGES ON DATABASE database\_name TO username;

* Autorização para qualquer tipo de objeto do SCHEMA:

GRANT USAGE ON SCHEMA schema\_name TO username;

GRANT ALL PRIVILEGES ON SCHEMA public TO michel;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema\_name TO username;

GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA schema\_name TO username;

* ROLE e USER comportam-se de maneiras semelhantes:

CREATE USER geronimo PASSWORD ‘123’;

ou

CREATE ROLE geronimo PASSWORD ‘123’;

Observação : sempre dê preferência a ROLE, pois USER está “deprecated”. É possível criar a role com determinadas permissões:

CREATE ROLE geronimo WITH LOGIN CREATEDB INHERIT PASSWORD 'brasil2';

* Arquivo de configuração: Os principais arquivos estão disponíveis na pasta “/etc/postgresql/{versão do bd}main.

marco@montreal ~ $ ls -l /etc/postgresql/9.5/main

-rw-r--r-- 1 postgres postgres 315 Ago 3 01:37 environment

-rw-r--r-- 1 postgres postgres 143 Ago 3 01:37 pg\_ctl.conf

-rw-r----- 1 postgres postgres 4639 Ago 10 09:51 pg\_hba.conf

-rw-r----- 1 postgres postgres 1636 Ago 3 01:37 pg\_ident.conf

-rw-r--r-- 1 postgres postgres 21625 Ago 3 01:37 postgresql.conf

-rw-r--r-- 1 postgres postgres 378 Ago 3 01:37 start.conf

* É possível alterar os valores das configurações indo diretamente nos arquivos, ou via linha de comando no prompt do postgresql:

postgres=# \d pg\_settings;

View "pg\_catalog.pg\_settings"

Column | Type | Modifiers

name | text |

setting | text |

unit | text |

category | text |

short\_desc | text |

extra\_desc | text |

context | text |

vartype | text |

source | text |

min\_val | text |

max\_val | text |

enumvals | text[] |

boot\_val | text |

reset\_val | text |

sourcefile | text |

sourceline | integer |

pending\_restart | boolean |

name: nome da variável ou GUC, como os desenvolvedores do postgres gostam de chamar;

setting: o valor atual no contexto atual para aquela variável. Note que o contexto pode variar de acordo com varias coisas, como a forma como o PostgreSQL foi compilado, parâmetros passados para o postmastar ao subir o serviço, configurações do postgresql.conf, base em que você está conectada, usuário conectado e finalmente, alterações realizadas na sessão atual;

unit: Algumas variáveis podem ter valores que remetem a unidades específicas como: ms (milisegundos), s (segundos), min (minutos), kB (kiloBytes) e 8kB. Preste a atenção nas unidades, particularmente quando o valor estiver em unidades de 8kB, pois o valor em setting, pode lhe confundir. Um valor de 256 com uma unidade de 8kB significa na verdade 256 x 8kB =  2mB;

category: As variáveis estão agrupadas em categorias, [vide a documentação](http://www.postgresql.org/docs/current/static/runtime-config.html). Fazer uma consulta apenas nas variáveis que estão na categoria que você procura pode ajudar a filtrar seus resultados ou ordenações.

short\_desc: Descrição sucinta sobre a variável;

extra\_desc: Descrição e detalhes adicionais sobre a variável. Não chega a substituir a documentação, mas pode lhe ajudar em alguns momentos;

context: Diz em qual tipo de situação a variável pode ser alterada. Os níveis mais baixos sempre podem ser alterados nos níveis mais baixos, mas não o contrário:

* + internal – não podem ser alteradas de jeito nenhum. Algumas podem ser alteradas durante a compilação do PostgreSQL. Ou seja, não adianta tentar sair alterando. Estão lá mais por referência.
  + postmaster – só podem ser alteradas após reiniciar o postgres. Você pode alterar parâmetros no postgresql.conf (ou usar o ALTER SYSTEM), mas as novas configurações só entrarão em vigor após reiniciar o postgres. Você também pode ajustar algumas variáveis passando parâmetros para o postmaster. Não recomendo fazer isso.
  + sighup – podem ser alterando com um reload (sinal SIGHUP ou usar a função [pg\_reload\_conf](http://www.postgresql.org/docs/9.4/static/functions-admin.html)).
  + backend – também podem serem alterados quando a conexão é estabelecida utilizando as variáveis de ambiente PGOPTIONS. No entanto uma vez iniciada a sessão, o valor configurado não pode ser mais alterado durante toda a vida desta sessão;
  + superuser – podem ser alterados com os comandos SET, ALTER ROLE e ALTER DATABASE, mas apenas super usuários podem executar estas alterações. Na prática isto significa que sessões de usuários que não são superusuários não podem ser alteradas no meio de uma sessão.
  + user – também podem ser alterados com SET, ALTER ROLE e ALTER DATABASE, mas não precisa ser um superusuário para fazê-lo.

vartype: sim senhores, os valores em no campo setting vem todos no formato texto… então você precisa saber que tipo de valor é permitido neste campo! As variáveis podem ser dos tipos  bool, enum, string, integer ou real.

source: Esta opção lhe diz de onde vem o valor configurado para aquela variável. Uma vez que dependendo do contexto, a variável pode ter sido ajustada em diversos contextos, ou apenas estar assumindo o valor padrão: o valor default no campo source significa que em nenhum momento esta variável foi alterada. Os possíveis valores para o campo source são: default,override, configuration file, environment variable, client, database ou user.

min\_val: valor mínimo permitido caso o tipo do valor seja numérico

max\_val: valor máximo permitido caso o tipo do valor seja numérico

enumvals: valores permitidos caso o tipo do valor seja enum

boot\_val: valor assumido pela variável ao iniciar o postgres se ela não for configurada em nenhum local

reset\_val: valor assumido pela variável se um RESET for solicitado no meio de uma sessão

sourcefile: nome do arquivo de configuração onde a variável foi ajustada. Só faz sentido utilizar este campo se você utilizar includes dentro do postgresql.conf

sourceline: linha do arquivo que alterou aquela variável.

* Para ler o conteúdo de cada parâmetro:

SELECT name FROM pg\_settings WHERE context = 'internal';

name

-----------------------

block\_size

data\_checksums

debug\_assertions

integer\_datetimes

lc\_collate

lc\_ctype

max\_function\_args

max\_identifier\_length

max\_index\_keys

segment\_size

server\_encoding

server\_version

server\_version\_num

wal\_block\_size

wal\_segment\_size

(15 rows)

#### context = ‘internal’

Uma variável cuja o contexto é internal, não pode ser alterado em nenhum momento, depois que o binário foi compilado.

#### context = ‘postmaster’

Agora, se tentarmos alterar um parâmetro cuja o contexto é postmaster, o valor só será alterado após reiniciar o postgres:

#### context = ‘user’

A única diferença aqui é que até mesmo um usuário comum pode alterar parâmetros com este contexto no meio da sua sessão.

* Para alterar o conteúdo do parâmetro podemos fazer da seguinte forma:

ALTER SYSTEM SET shared\_buffers = '256MB'; – Neste caso o parâmetro só valerá – após o restart do serviço

SHOW shared\_buffers;

* Para mais informações acesse o link abaixo:

<https://www.savepoint.blog.br/2015/01/14/aprenda-a-trabalhar-com-as-configuracoes-do-postgresql/>

Uma forma de liberar direitos de acesso:

CREATE ROLE chico WITH LOGIN PASSWORD ‘brasil2’;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema\_name TO PUBLIC

CREATE DATABASE itau;

\c itau

CREATE SCHEMA engenharia;

GRANT USAGE ON SCHEMA engenharia TO PUBLIC;

ALTER DEFAULT PRIVILEGES IN SCHEMA engenharia GRANT SELECT ON TABLES TO PUBLIC;

CREATE TABLE engenharia.projeto( codigoProjeto INT, descricao VARCHAR(50));

Observação: Com outro usuário tente dar um SELECT na tabela criada e verificar se teve sucesso.

Todo GRANT do item abaixo só terá validade para as tabelas existentes, novas tabelas não herdam a liberação de privilégios, a melhor maneira para pegar todas as tabelas seria com o comando ALTER DEFAULT e o GRANT SELECT… .

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema\_name TO PUBLIC

Primeiro deve-se fazer as configurações de liberação de acesso, só depois realizar os creates de tabelas. Para que tabelas criadas anteriores a criação do usuário

. Tablespace são criadas para armazenar num local especifico os objetos do database:

- - A localização só poderá ser uma que o usuário do postgres tenha todas as permissões

CREATE TABLESPACE producao LOCATION ‘/var/lib/postgres/9.5/’;

CREATE DATABASE producao TABLESPACE producao;

- - Podemos criar uma tabela especifica e colocá-la numa determinada tablespace

CREATE TABLE historico TABLESPACE producao\_historico;

* Criação de tabelas:

- - Se não for colocado o nome do Schema, a tabela será criada no Schema Public

CREAT TABLE {Schema}.{Tabela}(…) TABLESPACE {tablespace};