

Como Instalar e Utilizar o PostgreSQL no Ubuntu 16.04

Posted December 23, 2016 © 117.3k POSTGRESQL

By [Justin Ellingwood](#)

[Become an author](#)

Introdução

Sistemas gerenciadores de banco de dados são um componente chave em muitos sites web e aplicações. Eles fornecem uma forma estruturada de armazenar, organizar, e acessar informações.

PostgreSQL, ou Postgres, é um sistema gerenciador de banco de dados relacional que fornece uma implementação da linguagem de consulta SQL. Ele é uma escolha popular para muitos projetos pequenos e grandes e tem a vantagem de ser compatível com padrões e de possuir muitas características avançadas como transações confiáveis e concorrência sem bloqueios de leitura.

Neste guia, vamos demonstrar como instalar o Postgres em uma instância VPS Ubuntu 16.04 e seguir aprendendo algumas formas de utilizá-lo.

Instalação

Os repositórios padrão do Ubuntu contém os pacotes do Postgres, assim podemos instalá-lo facilmente utilizando o sistema de pacotes `apt`.

Como essa é nossa primeira vez utilizando o `apt` para essa seção, devemos iniciar atualizando nosso índice local de pacotes. Podemos então instalar o pacote Postgres e o pacote `-contrib` que adiciona algumas funcionalidades e utilitários extras.

to code, and get updates

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

Agora que nosso software está instalado, podemos passar a ver como ele funciona e como ele pode ser diferente de sistemas de gerenciamento de banco de dados similares que você já deve ter utilizado.

Utilizando Funções (roles) e Bancos de Dados do PostgreSQL

Por padrão, o Postgres utiliza um conceito chamado “roles” para tratar da autenticação e da autorização. Eles são, de alguma forma, similares a contas do estilo Unix regulares, mas o Postgres não distingue entre usuários e grupos e, em vez disso, prefere o termo mais flexível “role”.

Após a instalação o Postgres é configurado para utilizar a autenticação **ident**, o que significa que ele associa os roles Postgres como uma conta de sistema Unix/Linux relacionada. Se um role existe dentro do Postgres, um nome de usuário Unix/Linux com o mesmo nome será capaz de entrar com aquela função.

Existem algumas maneiras de utilizar essa conta para acessar o Postgres.

Alternando para a Conta do Postgres

O procedimento de instalação criou um usuário chamado `postgres` que é associado com o role padrão do Postgres. Para usar o Postgres, podemos fazer login nessa conta.

Altere para a conta `postgres` no seu servidor digitando:

```
$ sudo -i -u postgres
```

Agora você pode acessar o prompt do Postgres imediatamente digitando:

```
$ psql
```

Você será conectado e será capaz de interagir com o sistema de gerenciamento de banco de dados imediatamente.

Saia do prompt do Postgres digitando:

```
postgres=# \q
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

Acessando um prompt Postgres Sem Alternar Contas

Você também pode executar o comando que você gostaria com a conta `postgres` diretamente com `sudo`.

Por exemplo, no último exemplo, apenas queríamos chegar a um prompt do Postgres. Podemos fazer isso executando o comando simples `psql` como usuário `postgres` e com `sudo` como abaixo:

```
$ sudo -u postgres psql
```

Isso iniciará a sessão diretamente no Postgres sem o shell `bash` intermediário no meio.

Novamente, você pode sair da sessão interativa do Postgres digitando:

```
postgres=# \q
```

Criar uma Nova Função (Role)

Atualmente, temos apenas a função `postgres` configurada no banco de dados. Podemos criar novas funções através da linha de comando com o comando `createuser`. A flag `--interactive` irá solicitá-lo os valores necessários.

Se você estiver conectado com a conta `postgres`, você pode criar um novo usuário digitando:

```
$ createuser --interactive
```

Se, em vez disso, você preferir utilizar o `sudo` para cada comando sem alternar a partir da sua conta normal, você pode digitar:

```
$ sudo -u postgres createuser --interactive
```

O script solicitará algumas escolhas e, com base em suas respostas, execute os comandos corretos do Postgres para criar um usuário de acordo com suas especificações.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

```
Enter name of role to add: sammy
Shall the new role be a superuser? (y/n) y
```

Você pode ganhar mais controle ao passar flags adicionais. Confira as opções olhando para a página do manual:

```
$ man createuser
```

Criar um Novo Banco de Dados

Por padrão, outra suposição que o sistema de autenticação do Postgres faz é que haverá um banco de dados com o mesmo nome da função que está sendo usada para fazer o login, a que a função tem acesso.

Dessa forma, se na seção anterior nós criamos um usuário chamado **sammy**, essa função irá tentar se conectar ao banco de dados que também é chamado **sammy** por padrão. Você pode criar um banco de dados apropriado com o comando `createdb`.

Se você estiver conectado com a conta `postgres`, você digitaria algo como:

```
$ createdb sammy
```

Se, em vez disso, você preferir utilizar o `sudo` para cada comando sem alternar a partir da sua conta normal, você digitaria:

```
$ sudo -u postgres createdb sammy
```

Abrir um Prompt Postgres com a Nova Função

Para conectar-se com autenticação baseada em `ident`, você precisará de um usuário Linux com o mesmo nome que sua função Postgres e de banco de dados.

Se você não tiver um usuário Linux correspondente disponível, você pode criar um com o comando `adduser`. Você tem que fazer isso a partir de uma conta com privilégios `sudo` (não conectado como o usuário `postgres`):

```
$ sudo adduser sammy
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

Uma vez que você tenha uma conta apropriada disponível, você pode tanto alternar e conectar ao banco de dados digitando:

```
$ sudo -i -u sammy  
$ psql
```

Ou, você pode fazer isso em linha:

```
$ sudo -u sammy psql
```

Você será conectado automaticamente assumindo que todos os componentes foram configurados corretamente.

Se você quiser que seu usuário conecte-se a um banco de dados diferente, você pode fazer isso especificando o banco de dados, dessa forma:

```
$ psql -d postgres
```

Uma vez conectado, você pode verificar suas informações de conexão atuais digitando:

```
sammy=# \conninfo
```

Output

```
You are connected to database "sammy" as user "sammy" via socket in "/var/run/postg
```

Isso pode ser útil se você estiver se conectando a um banco de dados não-padrão ou com usuários não-padrão.

Criar e Deletar Tabelas

Agora que você sabe se conectar ao sistema de banco de dados PostgreSQL, podemos partir para saber como completar algumas tarefas básicas.

Primeiro, podemos criar uma tabela para armazenar alguns dados. Vamos criar uma tabela que descreve equipamentos de playground.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

```
CREATE TABLE table_name (  
    column_name1 col_type (field_length) column_constraints,  
    column_name2 col_type (field_length),  
    column_name3 col_type (field_length)  
);
```

Como você pode ver, nós demos um nome para a tabela, e depois definimos as colunas que queremos, bem como os tipos das colunas e o comprimento máximo do campo de dados. Também podemos opcionalmente adicionar restrições de tabela para cada coluna.

Você pode aprender mais sobre [como criar e gerenciar tabelas no Postgres aqui](#).

Para nossos propósitos, vamos criar uma tabela simples como essa:

```
CREATE TABLE playground (  
    equip_id serial PRIMARY KEY,  
    type varchar (50) NOT NULL,  
    color varchar (25) NOT NULL,  
    location varchar(25) check (location in ('north', 'south', 'west', 'east', 'nor  
    install_date date  
);
```

Fizemos uma tabela playground que faz o inventário dos equipamentos que possuímos. Ela começa com um ID de equipamento, que é do tipo `serial`. Esse tipo de dados é um inteiro com auto incremento. Demos a essa coluna a restrição de `primary key`, o que significa que os valores devem ser exclusivos e não nulos

Para duas de nossas colunas (`equip_id` e `install_date`), não definimos um comprimento de campo. Isso é porque alguns tipos de colunas não requerem um comprimento definido, porque o comprimento está implícito pelo tipo.

Em seguida temos as colunas para o tipo (`type`) e cor (`color`) do equipamento, cada um dos quais não pode estar vazio. Criamos a coluna de localização (`location`) e criamos uma restrição que requer que o valor seja um dos oito valores possíveis. A última coluna é uma coluna de data que grava a data em que instalamos o equipamento.

Podemos ver nossa nova tabela digitando:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ✕

Enter your email address

Sign Up

```
sammy=# \d
```

Output

```
              List of relations
Schema |          Name          | Type   | Owner
-----+-----+-----+-----
public | playground              | table  | sammy
public | playground_equip_id_seq | sequence | sammy
(2 rows)
```

Nossa tabela `playground` está aqui, mas também temos algo chamado `playground_equip_id_seq` que é do tipo `sequence`. Essa é a representação do tipo `serial` que demos à nossa coluna `equip_id`. Isso mantém o controle do próximo número na seqüência e é criado automaticamente para colunas desse tipo.

Se você quiser ver apenas a tabela sem a seqüência, você pode digitar:

```
sammy=# \dt
```

Output

```
              List of relations
Schema |   Name   | Type  | Owner
-----+-----+-----+-----
public | playground | table | sammy
(1 row)
```

Adicionar, Consultar e Excluir Dados em uma Tabela

Agora que temos uma tabela, podemos inserir alguns dados nela.

Vamos adicionar um escorregador (slide) e um balanço (swing). Fazemos isso chamando a tabela na qual queremos adicionar, nomeando as colunas e então, fornecendo os dados para cada coluna. Nosso escorregador e nosso balanço podem ser adicionados assim:

```
INSERT INTO playground (type, color, location, install_date) VALUES ('slide', 'blue', 'left', '2012-01-01')
INSERT INTO playground (type, color, location, install_date) VALUES ('swing', 'yellow', 'right', '2012-01-01')
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.

Sign Up

Você deve tomar cuidado ao inserir os dados para evitar alguns problemas comuns. Primeiro, tenha em mente que os nomes de colunas não devem estar entre aspas, mas os valores que você está inserindo para as colunas precisam estar entre aspas.

Outra coisa para ter em mente é que nós não inserimos um valor para a coluna `equip_id`. Isso é porque isso é gerado automaticamente quando uma nova linha é criada na tabela.

Podemos então recuperar as informações que adicionamos digitando:

```
sammy=# SELECT * FROM playground;
```

Output

```
equip_id | type  | color  | location  | install_date
-----+-----+-----+-----+-----
          1 | slide | blue   | south     | 2014-04-28
          2 | swing | yellow | northwest | 2010-08-16
(2 rows)
```

Aqui, você pode ver que nosso `equip_id` foi preenchido com sucesso e todos os outros dados foram organizados corretamente.

Se o escorregador no playground quebra e temos que removê-lo, podemos também remover a linha de nossa tabela digitando:

```
sammy=# DELETE FROM playground WHERE type = 'slide';
```

Se consultarmos nossa tabela novamente, veremos que nosso escorregador já não é mais parte da tabela.

```
sammy=# SELECT * FROM playground;
```

Output

```
equip_id | type  | color  | location  | install_date
-----+-----+-----+-----+-----
          2 | swing | yellow | northwest | 2010-08-16
(1 row)
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

Se quisermos modificar uma tabela depois que a mesma tiver sido criada para adicionar uma coluna a mais, podemos fazer isso facilmente.

Podemos adicionar uma coluna para mostrar a última visita de manutenção para cada equipamento digitando:

```
sammy=# ALTER TABLE playground ADD last_maint date;
```

Se você visualizar as informações de sua tabela novamente, você verá que uma nova coluna foi adicionada (mas nenhum dado foi inserido):

```
sammy=# SELECT * FROM playground;
```

Output

```
equip_id | type  | color  | location  | install_date | last_maint
-----+-----+-----+-----+-----+-----
          2 | swing | yellow | northwest | 2010-08-16   |
(1 row)
```

Podemos excluir uma coluna com a mesma facilidade. Se descobrimos que nossa equipe de trabalho usa uma ferramenta separada para acompanhar o histórico de manutenção, podemos nos livrar da coluna aqui digitando:

```
sammy=# ALTER TABLE playground DROP last_maint;
```

Como Atualizar Dados em uma Tabela

Sabemos como adicionar registros em uma tabela e como excluí-los, mas ainda não abordamos como modificar entradas existentes.

Você pode atualizar os valores de uma entrada existente consultando o registro desejado e definindo a coluna para o valor que você deseja usar. Podemos consultar o registro “swing” (isso irá corresponder a cada balanço na nossa tabela) e alterar sua cor para “red”. Isso pode ser útil se tivermos pintado o balanço:

```
sammy=# UPDATE playground SET color = 'red' WHERE type = 'swing';
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. 

Sign Up

Podemos verificar se a operação foi bem-sucedida consultando nossos dados novamente:

```
sammy=# SELECT * FROM playground;
```

Output

```
equip_id | type  | color | location | install_date  
-----+-----+-----+-----+-----  
        2 | swing | red   | northwest | 2010-08-16  
(1 row)
```

Como você pode ver, nosso balanço está agora registrado como sendo vermelho.

Conclusão

Você acabou de configurar o PostgreSQL no seu servidor Ubuntu 16.04. No entanto, ainda há muito mais a aprender com o Postgres. Aqui estão mais alguns guias que cobrem como usar o Postgres:

[A comparison of relational database management systems](#)

[Learn how to create and manage tables with Postgres](#)

[Get better at managing roles and permissions](#)

[Craft queries with Postgres with Select](#)

[Learn how to secure PostgreSQL](#)

[Learn how to backup a Postgres database](#)

By [Justin Ellingwood](#)

Translation: [Fernando Pimenta](#)

Was this helpful?

Yes

No



Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. 

Enter your email address

Sign Up

Related

TUTORIAL

How To Customize the PostgreSQL Prompt with psqlrc on Ubuntu 14.04

The psqlrc file customizes the behavior of the psql interactive command line...

TUTORIAL

Como Monitorar seu Banco de Dados PostgreSQL Gerenciado Usando o Nagios Core no Ubuntu 18.04

O autor escolheu o Free and Open Source Fund...

TUTORIAL

Como Analisar as Estatísticas do Banco de Dados PostgreSQL Gerenciado Usando o Elastic Stack no Ubuntu 18.04

O autor escolheu o Free...

TUTORIAL

How To Connect to a Managed Database on Ubuntu 18.04

If you're new to working with managed databases, the best way to perform...

Still looking for an answer?



Ask a question



Search for more help

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Enter your email address

Sign Up

2 Comments

Leave a comment...

Log In to Comment

^ BrunoDSouza March 4, 2017

1 Nice post, but

I had problems when I tried to run the '**spql**' command, or any other command, it follows the **error** message below:

```
psql: could not connect to server: No such file or directory
Is the server running locally and accepting
connections on Unix domain socket "/var/run/postgresql/.s.PGSQL.5432"?
```

how to solve this problems?

^ MrAcalf April 13, 2017

0 Same problem here, W8ting a solution



Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ✕

Enter your email address

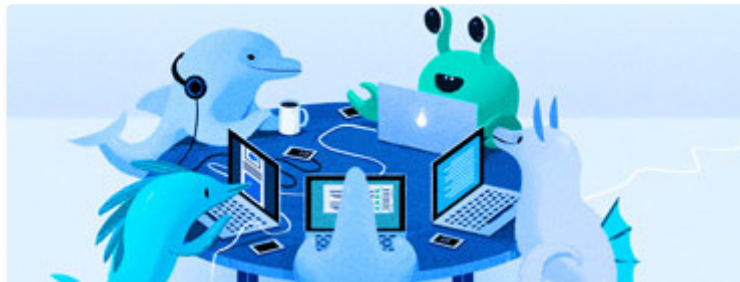
Sign Up

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



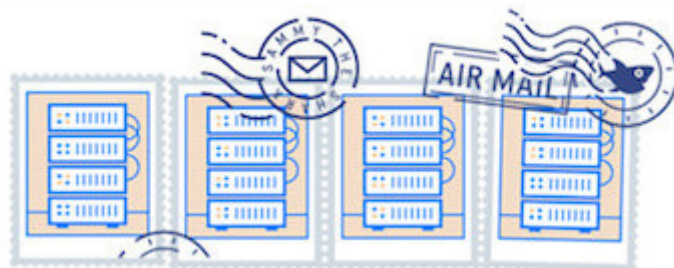
BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.



CONNECT WITH OTHER DEVELOPERS

Find a DigitalOcean Meetup near you.



GET OUR BIWEEKLY NEWSLETTER

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

Featured on Community [Intro to Kubernetes](#) [Learn Python 3](#) [Machine Learning in Python](#)
[Getting started with Go](#) [Migrate Node.js to Kubernetes](#)

DigitalOcean Products [Droplets](#) [Managed Databases](#) [Managed Kubernetes](#) [Spaces](#) [Object Storage](#)
[Marketplace](#)

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

[Learn More](#)



© 2019 DigitalOcean, LLC. All rights reserved.

Company

[About](#)
[Leadership](#)
[Blog](#)
[Careers](#)
[Partners](#)
[Referral Program](#)
[Press](#)
[Legal & Security](#)

Products

[Products Overview](#)
[Pricing](#)
[Droplets](#)
[Kubernetes](#)
[Managed Databases](#)
[Spaces](#)
[Marketplace](#)
[Load Balancers](#)

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.


Enter your email address

Sign Up



[Migrations](#)

- Community
- Contact
- Tutorials
- Support
- Q&A
- Sales
- Tools and Integrations
- Report Abuse
- Tags
- System Status
- Product Ideas
- Meetups
- Write for DOnations
- Droplets for Demos
- Hatch Startup Program
- Shop Swag
- Research Program
- Currents Research
- Open Source

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. 

Enter your email address

Sign Up