Preparando o ambiente para trabalhar com o Banco

Neste documento você terá o passo a passo para montar e configurar o nosso banco de dados **PostgreSQL** e **pgAdmin 4** em containers Docker.

Vamos ver como a montagem desse ambiente é rápida e simples.

A função deste material é deixar um ambiente com banco de dados, **PostgreSQL** e **pgAdmin 4**, pronto para ser utilizado pelo canal xxxx que dará dicas de Banco de Dados e linguagem SQL.

Esse material não tem a função de falar sobre o Docker, mas falaremos um pouco sobre a estrutura da solução.

Serão criados 2 (dois) containers, um com o banco de dados **PostgreSQL** e outro com **pgAdmin 4**, e uma network para a comunicação entre os containers.

Esse ambiente será criado utilizando-se o **Docker Compose**, um serviço do Docker para a criação e execução conjunta dos múltiplos containers de uma solução. Para isso vamos utilizar um arquivo chamado **docker-compose.yml** com os 3 (três) comandos para a criação e configuração dos containers e o network.

O nosso **docker-compose.yml** criará o ambiente de trabalho com os seguintes serviços:

Arquivo docker-compose.yml:

version: '3'

services:

  srv-bd-postgresql:

    image: postgres

    environment:

      POSTGRES\_PASSWORD: "curso"

    ports:

      - "15432:5432"

    volumes:

      - /home/luisjesus/Dev/Docker-Compose/PostgreSQL:/var/lib/postgresql/data

      networks:

      - srv-postgres-network

  srv-pgadmin:

    image: dpage/pgadmin4

    environment:

      PGADMIN\_DEFAULT\_EMAIL: "luisjesus.ti@gmail.com"

      PGADMIN\_DEFAULT\_PASSWORD: "curso"

    ports:

      - "16543:80"

    depends\_on:

      - srv-bd-postgresql

    networks:

      - srv-postgres-network

networks:

  srv-postgres-network:

    driver: bridge

Serão criados dois containers:

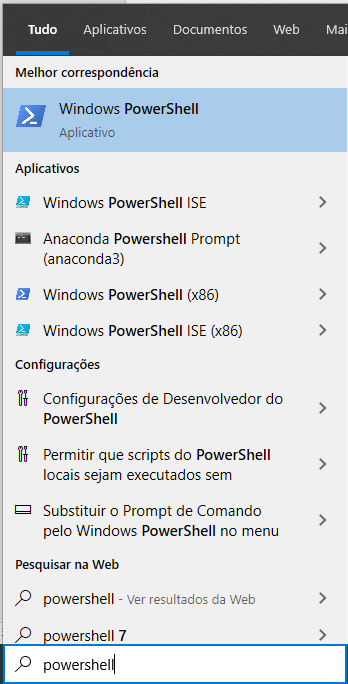
O primeiro refere-se ao banco de dados PostgreSQL: o serviço **srv-bd-postgresql** com porta de acesso 15432;

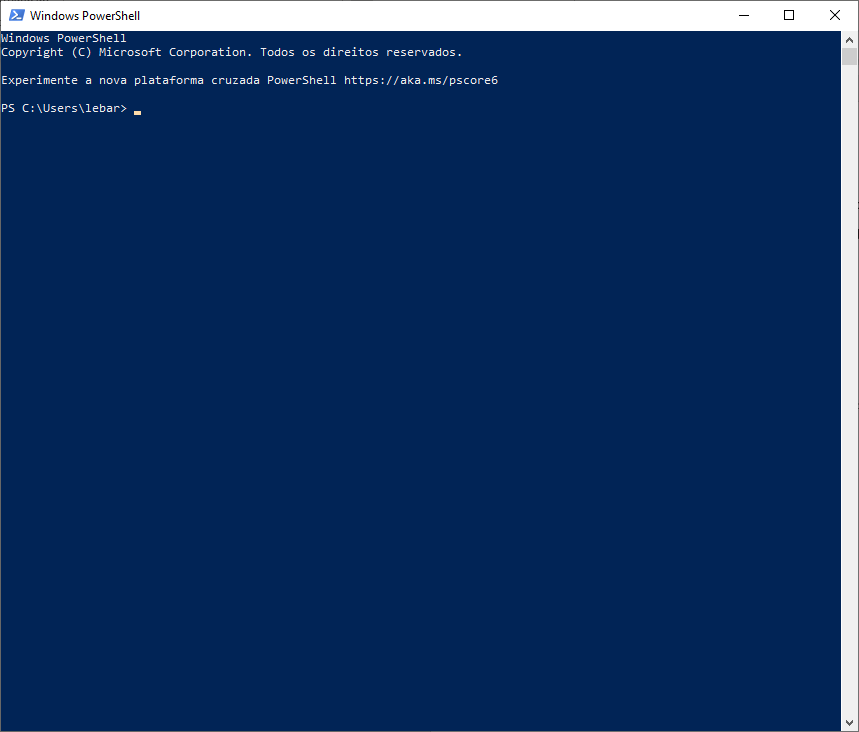
O segundo refere-se ao PGAdmin 4: o serviço **srv-pgadmin** com porta de acesso 16543;

O **srv-postgres-network** é o que permite a comunicação entre os containers **srv-pgadmin** e **srv-bd-postgresql**.

Agora vamos colocar para funcionar!

Vamos abrir o aplicativo **Windows PowerShell**.



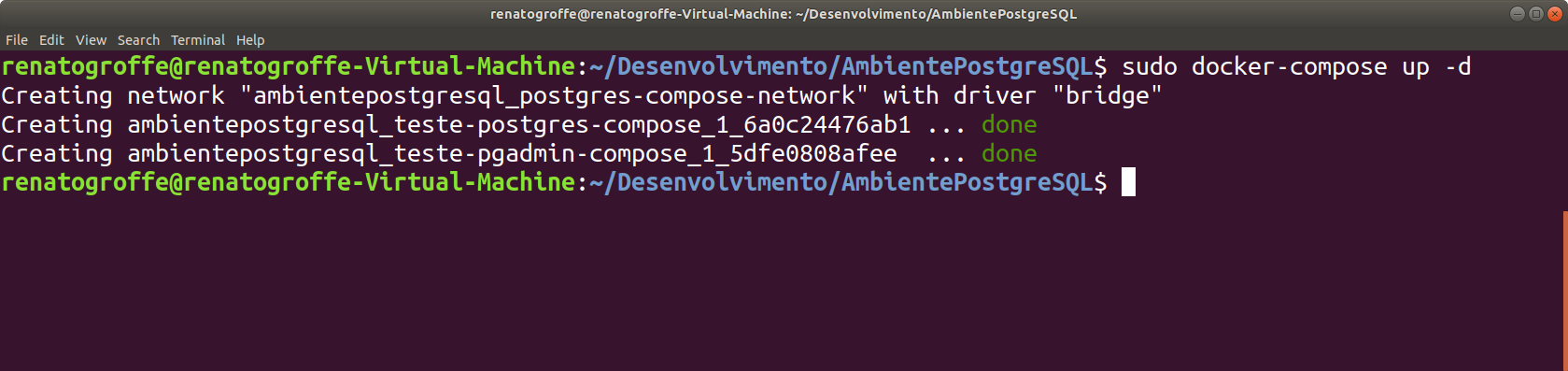


Vamos ao diretório onde está o arquivo docker-compose.yml e executar o comando **docker-compose up -d** vai criar os containers (vai fazer o download das imagens do PostgreSQL e do PGAdmin, caso ainda não existe na máquina).

-----

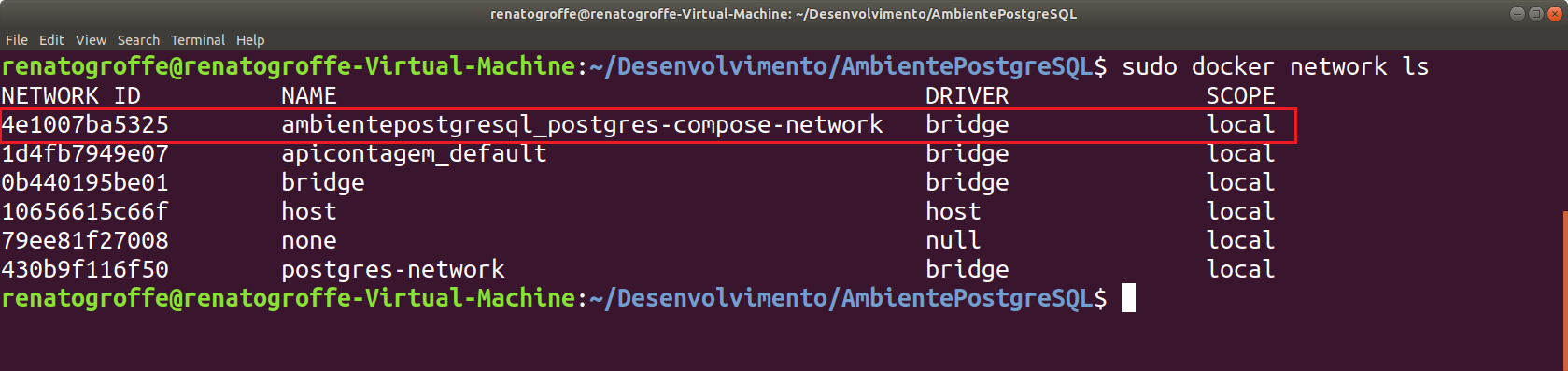
O comando **docker-compose up -d** procederá com a criação da network e dos containers esperados, efetuando inclusive o download das imagens se as mesmas ainda não existirem na máquina considerada (não foi o caso deste exemplo):





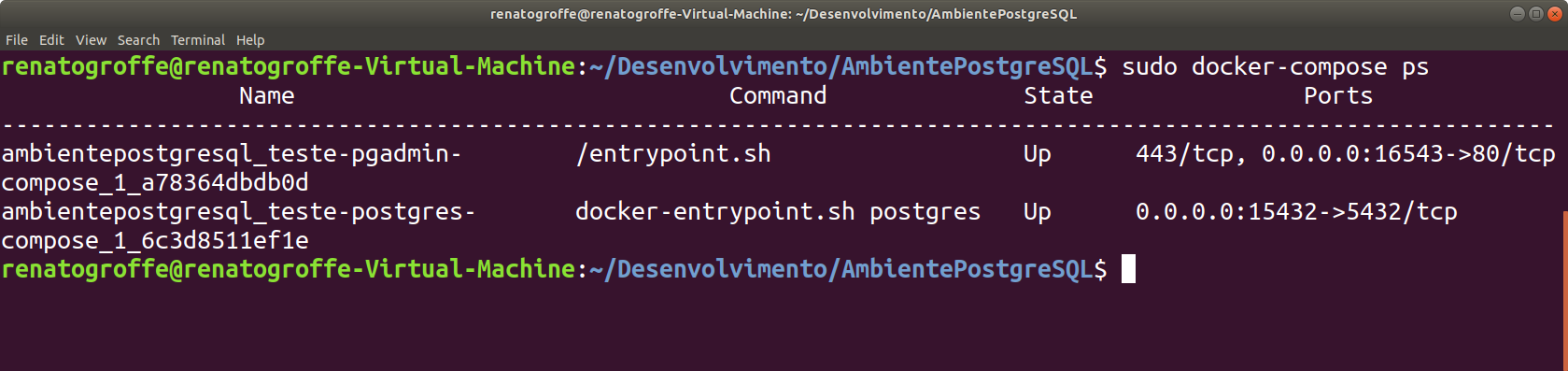
Com a instrução **docker network ls** podemos confirmar que a rede **postgres-compose-network** foi criada com sucesso (como **ambientepostgresql\_postgres-compose-network**, resultado da concatenação com o nome do diretório em que se encontra o arquivo **docker-compose.yml**):





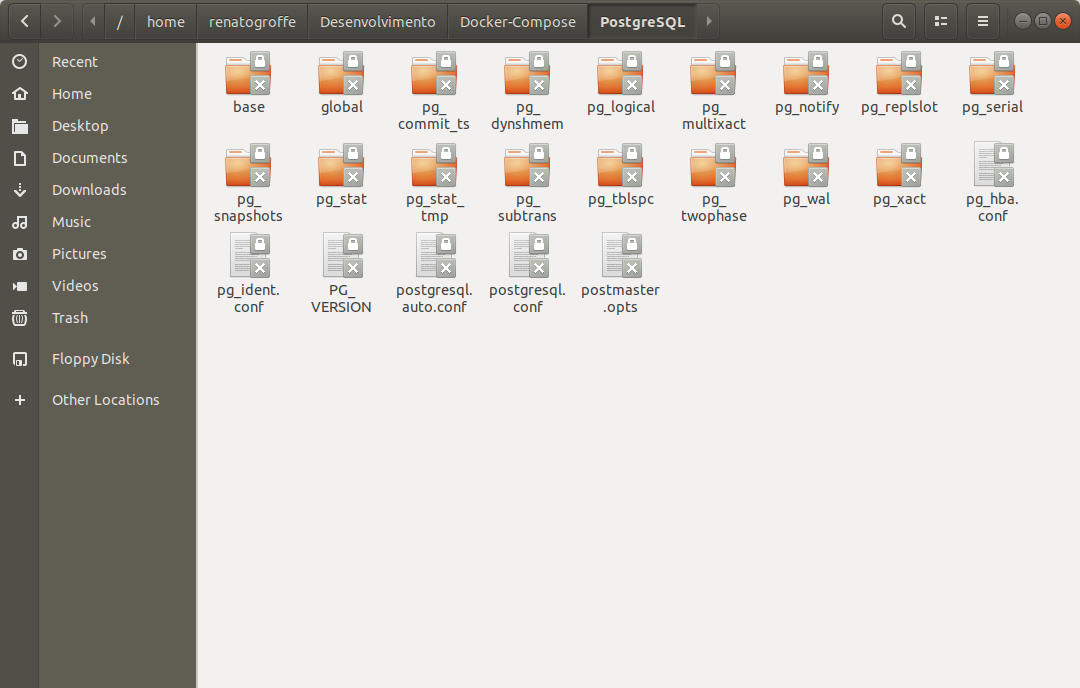
Já o comando **docker-compose ps** indicará que os containers do **PostgreSQL** (porta **15432**) e do **pgAdmin 4** (porta **16543**) foram gerados corretamente e se encontram em execução:





Na próxima imagem estão os arquivos e diretórios criados para o volume definido no arquivo **docker-compose.yml**:

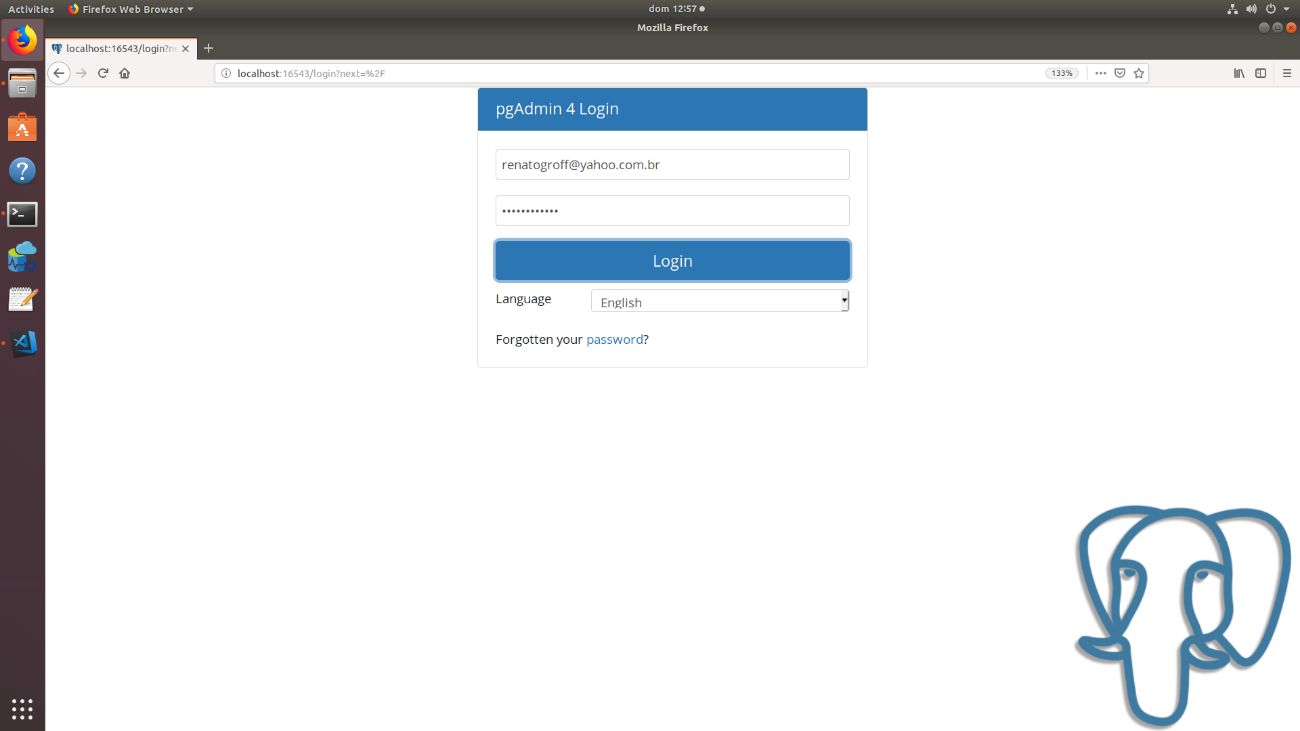




**Testando o ambiente**

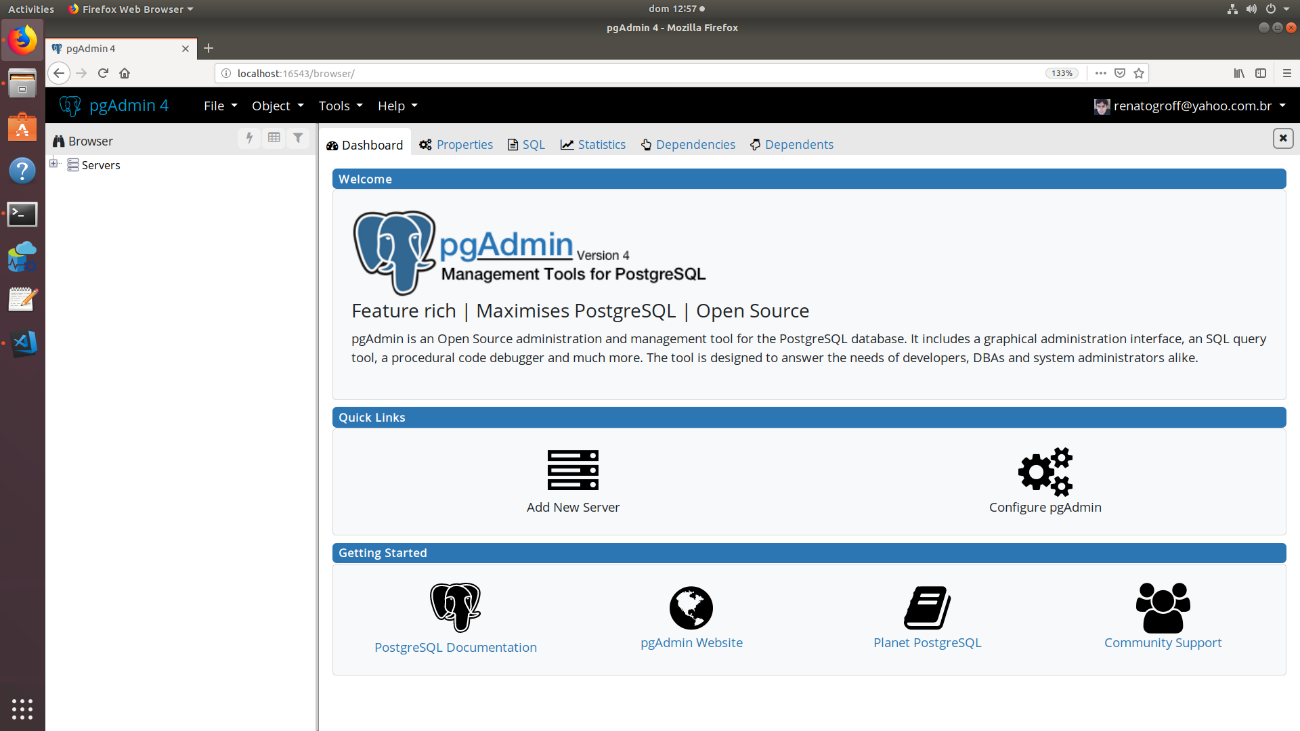
Um teste de acesso via browser ao **pgAdmin 4** (**http://localhost:16543**) exibirá a tela inicial desta solução:





Fornecendo as credenciais de acesso que estavam no arquivo **docker-compose.yml** aparecerá então o painel de gerenciamento do **pgAdmin 4**:

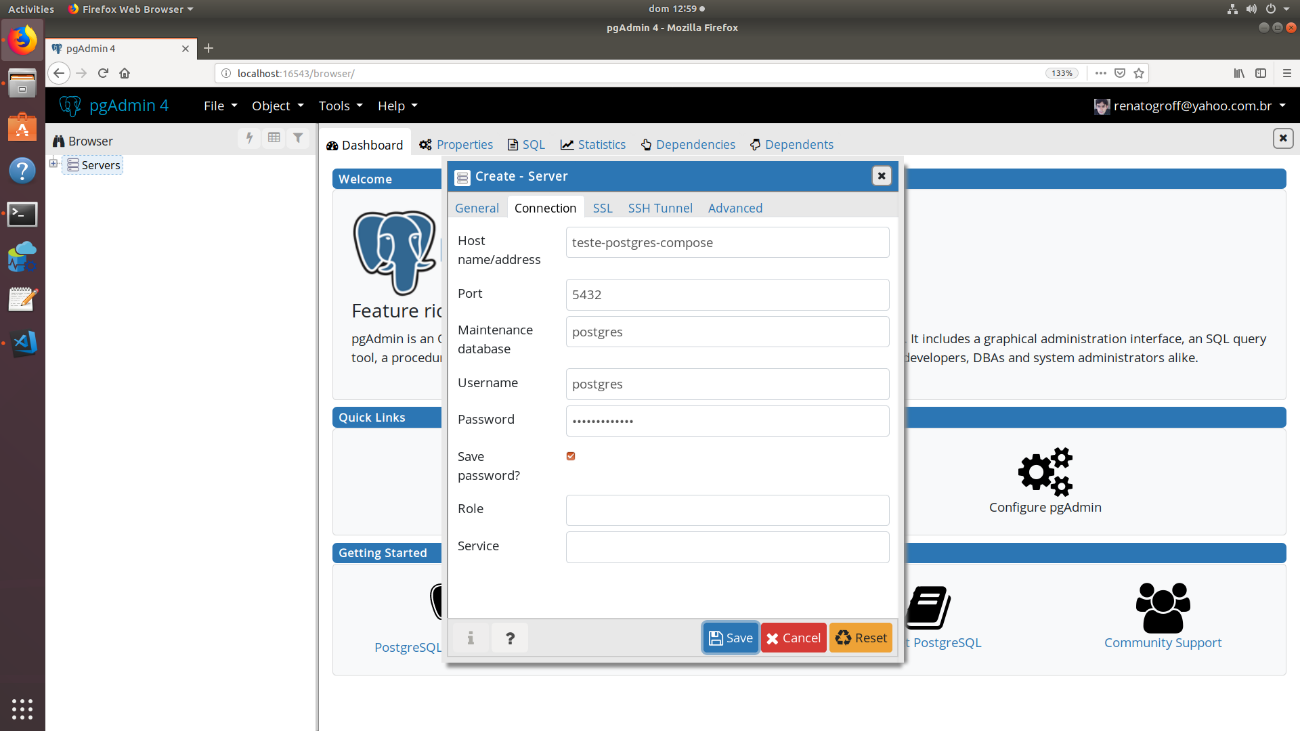




Ao criar a conexão para acesso à instância do **PostgreSQL** levar em conta as seguintes considerações:

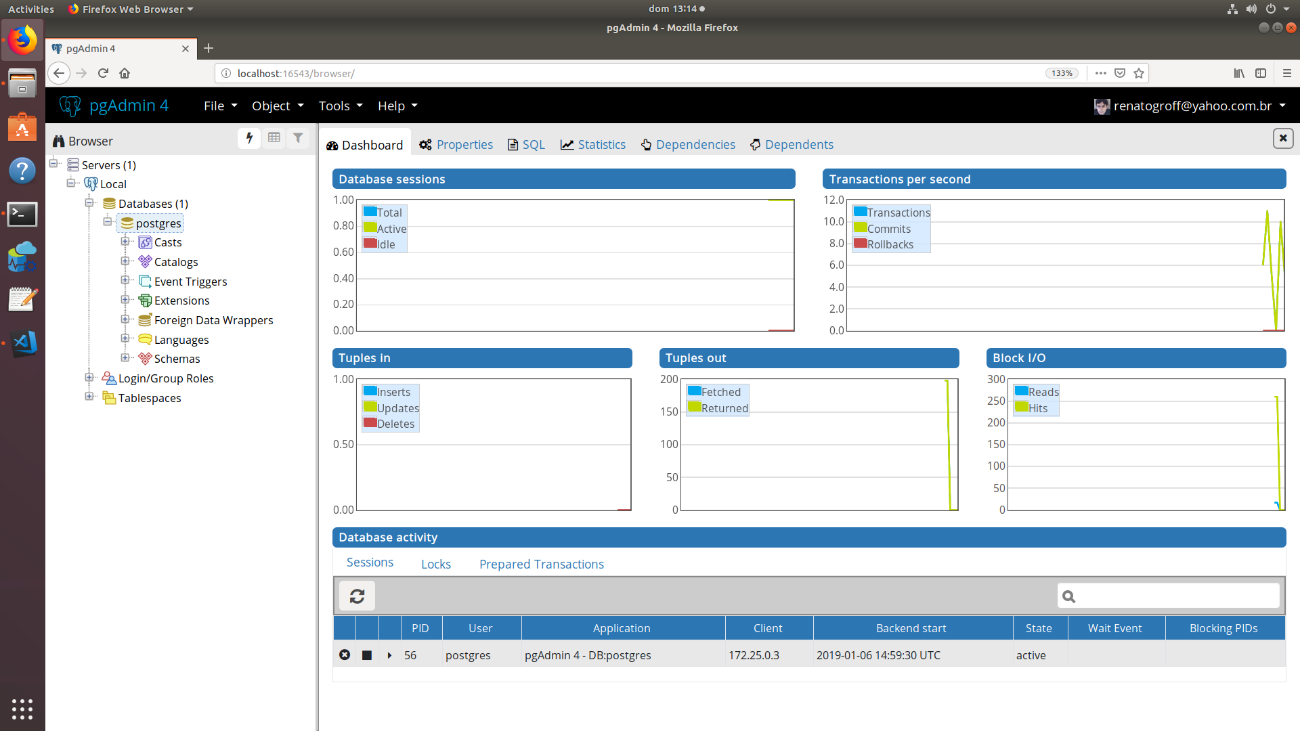
* Em **Host name/address** informar o nome do container que corresponde à instância do **PostgreSQL** (**teste-postgres-compose**);
* Em **Port** definir o valor **5432** (porta default de acesso ao container e disponível a partir da rede **postgres-compose-network**; não informar a porta em que o **PostgreSQL** foi mapeado no host);
* No atributo **Username** será informado o usuário default do **PostgreSQL** (**postgres**), bem como a senha correspondente em **Password** (**Postgres2019!**).





Na próxima imagem é possível observar que a conexão ao **PostgreSQL** via **pgAdmin 4** ocorreu com sucesso:





E aproveito este espaço para deixar aqui ainda um convite.

Dia **08/01/2019 (terça-feira) às 21h30 — horário de Brasília** — teremos o **primeiro hangout do ano de 2019** no **Canal .NET**. Desta vez receberemos o **MVP Luiz Carlos Faria**, que fará uma apresentação justamente sobre o **uso do Portainer como solução de administração e troubleshooting de containers Docker**.

Para efetuar a sua inscrição acesse a [**página do evento no Meetup**](https://www.meetup.com/pt-BR/dotnet-Sao-Paulo/events/257336222/). A transmissão acontecerá via **YouTube**, em um link a ser divulgado em breve.

**Referências**

[**Docker para Desenvolvedores .NET - Guia de Referência**](https://medium.com/@renato.groffe/docker-para-desenvolvedores-net-guia-de-refer%C3%AAncia-6f9bad2c244e)

[**PostgreSQL - Docker Hub**](https://hub.docker.com/_/postgres/)

[**pgAdmin 4 - Docker Hub**](https://hub.docker.com/r/dpage/pgadmin4/)