



Deploy web applications with a database to Render

The goal of this exercise is to deploy the same [PHP Todolist](#) application as in previous exercises, but this time on the Render Platform-as-a-service (PaaS) cloud instead of your own server in the Infrastructure-as-a-Service (IaaS) Microsoft Azure Web Services cloud. This illustrates the difference between the two cloud service models.

This guide assumes that you are familiar with [Git](#) and that you have a basic understanding of what a Platform-as-a-Service is.



Work on your local machine, NOT your cloud server. The goal of this exercise is to deploy on [Render](#), not your own server, to illustrate the difference between Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS).

Table of contents

- [Legend](#)
- [Install PostgreSQL](#)
 - [macOS](#)
 - [Windows](#)
- [Getting your Todolist fork up-to-date](#)
 - [Add the upstream as a remote](#)
 - [Fetch data from upstream](#)
 - [Push the new branch to GitHub](#)
- [Create and configure a PostgreSQL Database on Render](#)
 - [Create a Render account](#)
 - [Create a PostgreSQL instance](#)
 - [Connect to the database and create tables](#)
- [Deploy the application](#)

-  Create a web service
-  Define environment variables
-  Deploy the web service
-  What have I done?

Legend

Parts of this exercise are annotated with the following icons:

-  A task you **MUST** perform to complete the exercise
-  An optional step that you may perform to make sure that everything is working correctly, or to set up additional tools that are not required but can help you
-  The end of the exercise
-  The architecture of the software you ran or deployed during this exercise.
-  Troubleshooting tips: how to fix common problems you might encounter

Install PostgreSQL

PostgreSQL is a relational database management system that is very similar to MySQL. We use it here because we can deploy it with one click on Render. Other benefits of using PostgreSQL are performance, concurrency and SQL language support. You will need to install PostgreSQL on your own machine in order to access the remote instance hosted on Render using the `psql` command-line interface. The installation procedure differs on macOS and Windows.

macOS

To install PostgreSQL, you will be using Homebrew, the leading package manager for Mac. You may install it directly from your terminal, by entering:

```
$> /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/)
```

Once this is done, you can easily install packages by writing `brew install` followed by the name of the package:

```
$> brew install postgresql@18
```

Check that you have access to the `psql` command by entering:

```
$> psql --version  
psql (PostgreSQL) 18.x (Homebrew)
```

Windows

Go to the [PostgreSQL downloads page](#) and choose version **18.x for Windows x86-64**. Launch the installer and follow the installation instructions. You can decide to install **only** the command-line tools. The following instructions assume you installed PostgreSQL in the default directory on your C drive.

The installer does not take care of adding `psql` to your shell's path. You will therefore add it manually. Open Git Bash and enter the following commands:

```
$> echo 'export PATH=$PATH:"/c/Program Files/PostgreSQL/18/bin/' >> ~/.bashrc  
$> source ~/.bashrc
```

Check that you have access to the `psql` command by entering:

```
$> psql --version  
psql (PostgreSQL) 18.x (Homebrew)
```

! Getting your Todolist fork up-to-date.

When you started working on the Todolist application, you forked an existing codebase from a GitHub repository. While you were working on your configuration, the team with access to the original repository implemented the changes necessary for a PaaS deployment in a branch called `docker-postgres`.

By default, your fork does not track changes from the original repo, which is also commonly referred to as the **upstream**. Let's reconfigure our repository so that it can fetch data from there.

Tip

If you do not remember where the Todolist repository is stored on your local machine, you can simply clone it again from GitHub by running `git clone git@github.com:JohnDoe/php-todo-ex.git`. Don't forget to replace `JohnDoe` with your GitHub username.

! Add the upstream as a remote

From the terminal, move into your repository and add the upstream repository as a remote (this time, leave `ArchidDep` in the URL, you *want* to use the original URL and not your own):

```
$> cd php-todo-ex
$> git remote add upstream https://github.com/ArchidDep/php-todo-ex
```

More information

Unlike the [automated deployment exercise](#), you will not be pushing to this remote. You couldn't anyway, as you are not a collaborator on the upstream repository so you do not have the right to push. Instead, you will use it to fetch up-to-date code from a branch.

! Fetch data from upstream

Fetch all commits from the upstream repository:

```
$> git fetch upstream
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 11 (delta 4), reused 11 (delta 4), pack-reused 0
Unpacking objects: 100% (11/11), 3.20 KiB | 545.00 KiB/s, done.
From https://github.com/ArchiDep/php-todo-ex
* [new branch]      docker-postgres -> upstream/docker-postgres
* [new branch]      main          -> upstream/main
```

As you can see, this gives you access to upstream branches, including one called `upstream/docker-postgres`. With the next command you will copy the content of that upstream branch into your own branch.

```
$> git switch -c docker-postgres upstream/docker-postgres
branch 'docker-postgres' set up to track 'upstream/docker-postgres'.
Switched to a new branch 'docker-postgres'
```

This command will create a new branch in **your** local repository, based on the contents of the upstream branch. This command automatically switches you to the new branch. If you browse through the project in a code editor or by using `cat`, you should now be able to see changes to `todolist.sql`, as well as a mysterious new `Dockerfile`.

More information

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. A Dockerfile is a text file that contains instructions for how to build a

Docker image. We will learn more about Docker later in the course, and you can of course learn more [on the Docker website](#).

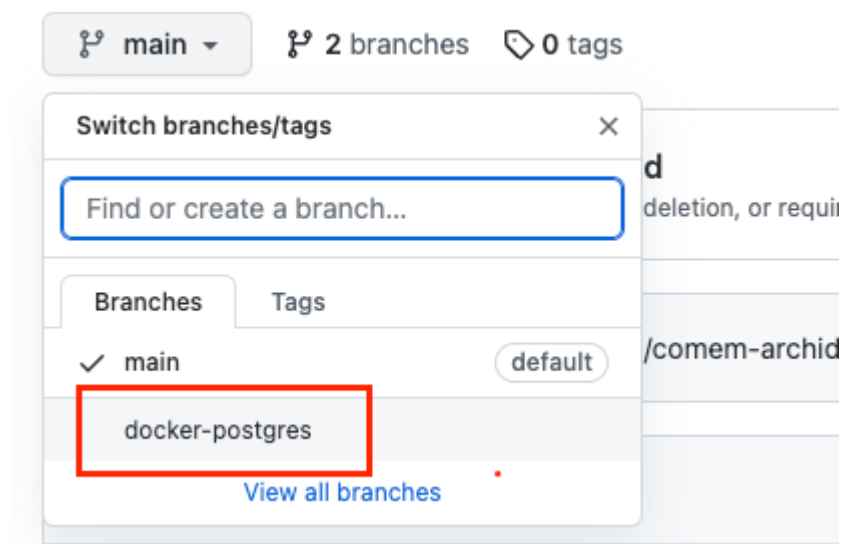
! Push the new branch to GitHub

```
$> git push origin
```

```
...
```

```
* [new branch]      docker-postgres -> docker-postgres
```

You can go check on GitHub whether your new branch has been pushed, by displaying the branch dropdown:



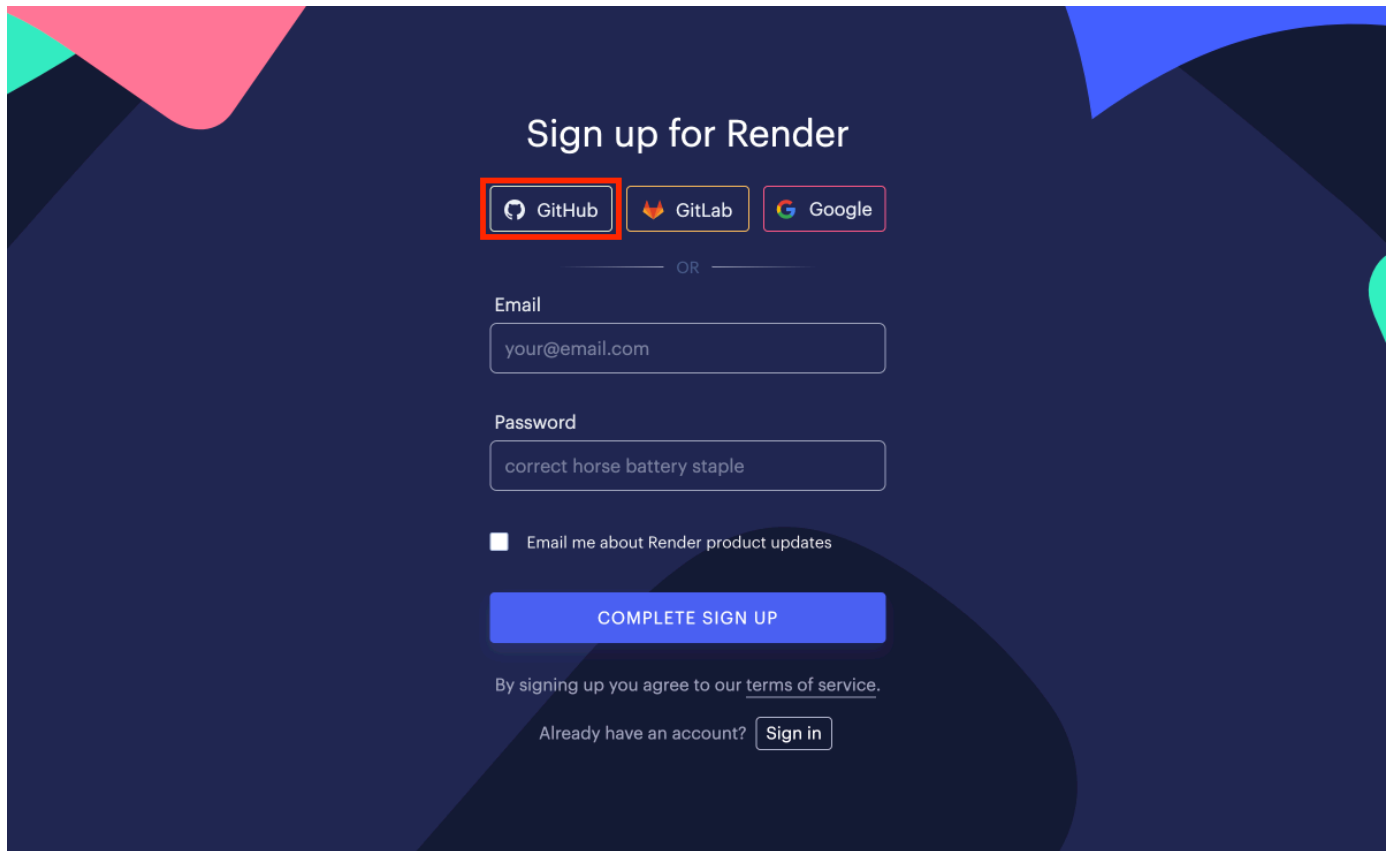
💡 Let's note that this whole step has nothing to do with PaaS deployments in and of themselves. It is just a corollary of some code changes that had to be made for the Todoist to work with PostgreSQL and Docker.

! Create and configure a PostgreSQL Database on Render

Instead of manually configuring a Linux server, you will be provisioning a couple of services on Render. The first is a PostgreSQL Database.

! Create a Render account

Start by creating a new Render account. If you choose to register using GitHub, you will be able to skip linking these two accounts together later:



The screenshot shows the 'Sign up for Render' page. At the top, the title 'Sign up for Render' is centered. Below it are three buttons for social login: GitHub, GitLab, and Google. The GitHub button is highlighted with a red rectangular box. Below these buttons is a horizontal line with the word 'OR' in the center. Underneath is an 'Email' field with the placeholder text 'your@email.com'. Below the email field is a 'Password' field with the placeholder text 'correct horse battery staple'. Below the password field is a checkbox labeled 'Email me about Render product updates'. Below the checkbox is a blue button labeled 'COMPLETE SIGN UP'. Below the button is the text 'By signing up you agree to our [terms of service](#).' At the bottom, there is a link 'Already have an account?' followed by a 'Sign in' button.

! Create a PostgreSQL instance

Sign-in to your Render account and click the **new PostgreSQL** button:

render

DashboardBlueprintsEnv GroupsDocsCommunityHelp

New +

Overview

Get up and running in minutes



Static Sites

Static sites are automatically served over a global CDN. Add a custom domain and get free, fully-managed SSL. [?](#)

New Static Site



Web Services

Web services include zero-downtime deploys, persistent storage and PR previews. Scale up and down with ease. [?](#)

New Web Service



Private Services

Private services are only accessible within your Render network and can speak any protocol. [?](#)

New Private Service



Background Workers

Background workers are suitable for long running processes like consumers for queues and streaming. [?](#)

New Worker



Cron Jobs

With cron jobs you can schedule any command or script to run on a regular interval. [?](#)

New Cron Job



PostgreSQL

Fully-managed hosted PostgreSQL with internal and external connectivity, and automated daily backups. [?](#)

New PostgreSQL



Redis

A cloud based in-memory key value datastore. Render offers fully managed hosted Redis instances. [?](#)

New Redis



Blueprints

A Blueprint specifies your Infrastructure as Code in a single file. Use it to set up all your services at once. [?](#)

New Blueprint

Warning

You can only have 1 active PostgreSQL deployment in the free Render tier. If you want more, you gotta pay.

This will take you to the following setup page, where you will need to configure:

- A name for your deployment
- A name for the database
- A username
- The region where the database is deployed (pick the one closest to your customers).

A password will be automatically generated for you.

render

Dashboard

Blueprints

Env Groups

Docs

Community

Help

New +

New PostgreSQL

Name

Database

User

Region

PostgreSQL Version

Datadog API Key

todolist

todolist

todolist

Frankfurt (EU Central)

15

Please [enter your payment information](#) to select an instance type with higher limits.

Instance Type	RAM	CPU	Storage	Price
☒ Free	256 MB	Shared	1 GB	\$0 / month
☐ Starter	256 MB	Shared	1 GB	\$7 / month
☐ Standard	1 GB	1 CPU	16 GB	\$20 / month
☐ Pro	4 GB	2 CPU	96 GB	\$95 / month
☐ Pro Plus	8 GB	4 CPU	256 GB	\$185 / month

Need a [custom plan](#)? We support up to 512 GB RAM, 64 CPUs, and 5 TB storage.

Free databases will expire in 90 days and will be deleted if not upgraded. [Learn more about free instance type limits.](#)

Create Database

When you are done, click **Create Database** and your PostgreSQL database will be provisioned automatically. Be patient, this process can take a few minutes. Once it is deployed you will be taken to a page with information pertaining to your new database and you should see the following:

Status

Available

! Connect to the database and create tables

At this point, you have a database. Congratulations. But you still need to set its tables up. As you did in the first Todolist tutorial, you will be running the `todolist.sql` script on the database, albeit remotely.

More information

The script is a bit different than the previous one because of two factors. First, we are using PostgreSQL instead of MySQL. Second, we do not need to create a database. As a matter of fact, this script is a bit simpler than the previous one.

Go back to your terminal and make sure you are in your repository and on the `docker-postgres` branch:

```
$> git branch --show-current  
docker-postgres
```

If not, check out the correct branch with the `git switch docker-postgres` command.

Next, connect to your PostgreSQL database from the command line. On the Render dashboard, you should be able to see a **Connections** section. This is where all the connection information to your database lives. You will need this information more than once, so keep this tab open.

The information you need to connect to the database shell is located in the **PSQL Command** field. You can display or copy the contents of this field by clicking the icons to the left of the hidden characters.

Connections

Hostname ⓘ dpg-cehh67g2i3mqvl9i29sg-a

Port 5432

Database todolist_45ig

Username todolist

Password



.....

Internal Database URL



.....

External Database URL



.....

PSQL Command



.....

Copy and paste the command in your terminal. This will connect you directly to the remote database deployed by Render.

```
$> PGPASSWORD=your_password psql -h your_host.frankfurt-postgres.render.com -U you
psql (14.6 (Homebrew), server 15.1)
WARNING: psql major version 16, server major version 16.
        Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_128_GCM_SHA256, bits: 128, com
Type "help" for help.
```

```
your_database=>
```

You can now execute the `todolist.sql` file:

```
your_database=> \i todolist.sql
```

```
CREATE TABLE
```

Tip

You can make sure the script worked by displaying all the `todo` table's columns:

```
your_database=> \d+ todo
```

Column	Type	Default
<code>id</code>	<code>integer</code>	<code>nextval('todo_id_seq'::regclass)</code>
<code>title</code>	<code>character varying(2048)</code>	
<code>done</code>	<code>boolean</code>	<code>false</code>
<code>created_at</code>	<code>timestamp without time zone</code>	<code>CURRENT_TIMESTAMP</code>

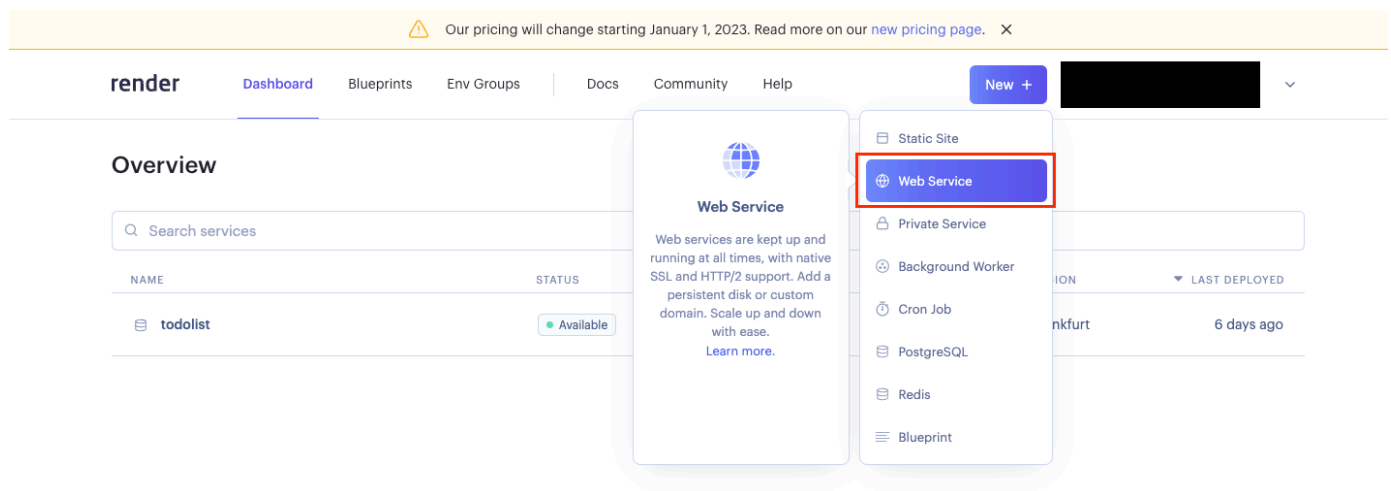
Now quit the PostgreSQL shell by entering `\q`.

! Deploy the application

Now that you have a database in place, it is time to deploy the web application itself.

! Create a web service

From your Render dashboard, hover over the bluish **“New”** button and select **Web Service**.



Render web services need to be connected to a Git repository hosted either on GitHub or GitLab. This step will allow you to automate deployments from your codebase. Instead of manually setting up hooks like in the [Automated Deployment exercise](#), you will rely on Render to take care of this for you.



Similar to GitHub, [GitLab][Gitlab] is also a version control platform that allows developers to manage and track changes to their codebase. They both use the Git version control system. Although they share the majority of their feature sets, GitLab can be self-hosted, which means that you can install and run it on your own servers. This can be useful for organizations that want to have more control over their infrastructure or that have specific security or compliance requirements.

If you are signed up using GitHub, you should see a list of all the repositories that can be used to create a web service. If not, you will need to follow the procedure to link your GitHub account to Render. Choose the appropriate repository for the purposes of this deployment and click **connect**.

render [Dashboard](#) [Blueprints](#) [Env Groups](#) [Docs](#) [Community](#) [Help](#) [New +](#)

Create a new Web Service

Connect your Git repository or use an existing public repository URL.

Connect a repository

  [Connect](#)

  [Connect](#)

  [Connect](#)

  [Connect](#)

  [Connect](#)

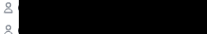
  [Connect](#)

Public Git repository

Use a **public repository** by entering the URL below. Features like [PR Previews](#) and [Auto-Deploy](#) are not available if the repository has not been configured for Render.

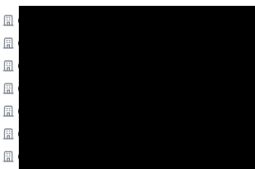
[Continue](#)

GitHub




[Configure account](#)

GitLab



[Configure account](#)

More information

As you can see, you can connect any public Git repository to Render by entering an URL in the field.

Once you have connected the repository, you will need to configure the deployment. Make sure you set the following basic options up:

- A name for your web service.
- The region where the service is deployed (pick the one closest to your customers).
- The branch from your repository that should be deployed (`docker-postgres`).
- The runtime environment (should automatically have Docker selected).
- The pricing tier.

You are deploying a web service for [simonpinkas/comem-archidep-php-todo-exercise](#).

Name

A unique name for your web service.

todo-render

Region

The [region](#) where your web service runs. Services must be in the same region to communicate privately and you currently have services running in **Frankfurt**.

Frankfurt (EU Central)

Branch

The repository branch used for your web service.

docker-postgres

Root Directory Optional

Defaults to repository root. When you specify a [root directory](#) that is different from your repository root, Render runs all your commands in the [specified directory](#) and ignores changes outside the directory.

e.g. src

Environment

The runtime environment for your web service.

Docker

Please [enter your payment information](#) to select an instance type with higher limits.

Instance Type	RAM	CPU	Price
☒ Free	512 MB	Shared	\$0 / month
☐ Starter	512 MB	0.5 CPU	\$7 / month

! Define environment variables

In addition to these basic options, we will directly set up our environment variables on this page. Scroll down a bit and click the **Advanced** button. From there, you can add an arbitrary amount of environment variables. You will use the following ones to connect your application to the PostgreSQL database you created earlier. All of the values can be found in the connection panel of your database's dashboard:

Environment variable

Description

DB_HOST

The host at which the PostgreSQL database can be reached.

DB_PORT

The port at which the PostgreSQL database can be reached.

DB_NAME

The name of the PostgreSQL database.

Environment variable	Description
DB_USER	The PostgreSQL user to connect as.
DB_PASS	The password of the PostgreSQL user.

Our pricing will change starting January 1, 2023. Read more on our [new pricing page](#). ✕

render Dashboard Blueprints Env Groups Docs Community Help New + 

Advanced ✕

Use environment variables to store API keys and other configuration values and secrets. You can access them in your code like regular environment variables, for example with `os.getenv()` in Python or `process.env` in Node.

DB_HOST		
DB_PORT	5432	
DB_NAME		
DB_USER		
DB_PASS		

Add Environment Variable

More information

You can also store secret files (like `.env` or `.npmrc` files and private keys) in Render. These files can be accessed during builds and in your code just like regular files. You can upload them right in this configuration panel or from the service's dashboard, post-deployment.

! Deploy the web service

Once you are done configuring your deployment, you may click the **Create Web Service** button at the bottom of the page. This will take you to the deployment page, where you will be able to follow along the logs and discover the domain Render has attributed to your app.

Our pricing will change starting January 1, 2023. Read more on our [new pricing page](#). X

render Dashboard Blueprints Env Groups Docs Community Help New +

WEB SERVICE

todo-render Docker Free Plan simonpinkas/comem-archidep-php-todo-exercise docker-postgres Connect Manual Deploy

https://todo-render-c5cp.onrender.com

Events Builds too slow? Upgrade to a paid plan to go faster. Learn more about free instance type limits.

Logs December 27, 2022 at 3:03 PM Live

Disks 20194ae Update Dockerfile

Environment

Shell Search logs Search Maximize Scroll to top

```
Dec 27 03:04:23 PM 2022/12/27 14:04:23 [info] 20#20: *2 client closed connection while waiting for request, client: 127.0.0.1, server: 0.0.0.0:80
Dec 27 03:04:23 PM 2022/12/27 14:04:23 [info] 21#21: *3 client closed connection while waiting for request, client: 127.0.0.1, server: 0.0.0.0:80
Dec 27 03:04:23 PM 2022/12/27 14:04:23 [info] 22#22: *4 client closed connection while waiting for request, client: 127.0.0.1, server: 0.0.0.0:80
Dec 27 03:04:23 PM [27-Dec-2022 14:04:23] NOTICE: fpm is running, pid 17
Dec 27 03:04:23 PM [27-Dec-2022 14:04:23] NOTICE: ready to handle connections
Dec 27 03:04:24 PM 2022/12/27 14:04:24 [info] 19#19: *5 client closed connection while waiting for request, client: 127.0.0.1, server: 0.0.0.0:80
Dec 27 03:04:24 PM 2022-12-27 14:04:24,274 INFO success: php-fpm entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
Dec 27 03:04:24 PM 2022-12-27 14:04:24,274 INFO success: nginx entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
```

Scroll to bottom

Feedback Invite a Friend Contact Support

Once the deployment has succeeded, you will be able to visit the todo list at the URL provided by Render. You may also use a custom domain by following [this tutorial](#).

Warning

This is a free service, so there are some obvious limitations.

First, deploys are sloooooow. Second, bandwidth and running hours are limited.

Third, your service will shut down if there is no activity for more than 15 minutes:

This can cause a response delay of up to 30 seconds for the first request that comes in after a period of inactivity.

Learn more about the limits of free Render accounts [here](#).

What have I done?

A whole lot! By using Render, GitHub and Docker, you automated a bunch of things that were done manually in the previous exercises. Here's what was configured for you:

- Process management with Docker & PHP-FPM
- Reverse proxying with nginx
- TLS/SSL encryption with Let's Encrypt
- Automated deployment

But this isn't magic, it's building on the work of others:

- First, there's the Dockerfile. It may not seem like a whole lot, but if you look at the first line, you might notice that we are importing something from [richarvey/nginx-php-fpm](https://github.com/richarvey/nginx-php-fpm). This is actually a popular (and fairly complex) Dockerfile build by somebody else. This is what automatically sets up PHP-FPM and nginx for us.
- Second, there's Render: despite its limitations in the free tier, we are getting free hosting, automated deployments and encryption.
- Finally, there's GitHub whose API allows the connection between your repo and Render to be very very easily configured.

Most of the technology and software that we have used throughout this course has been made possible by the contributions of others in the open community. Consider how you can contribute to open source projects by submitting code, writing documentation or reporting bugs and issues.