

# Parking Status Configuration Guide

This is a step-by-step guide on how to configure Parking Status to run on your local machine. In this guide we will:

1. Explain necessary prerequisites
2. Clone the project from GitHub
3. Configure the PostgreSQL database
4. Configure and Run the Spring Boot backend
5. Configure and Run the React frontend

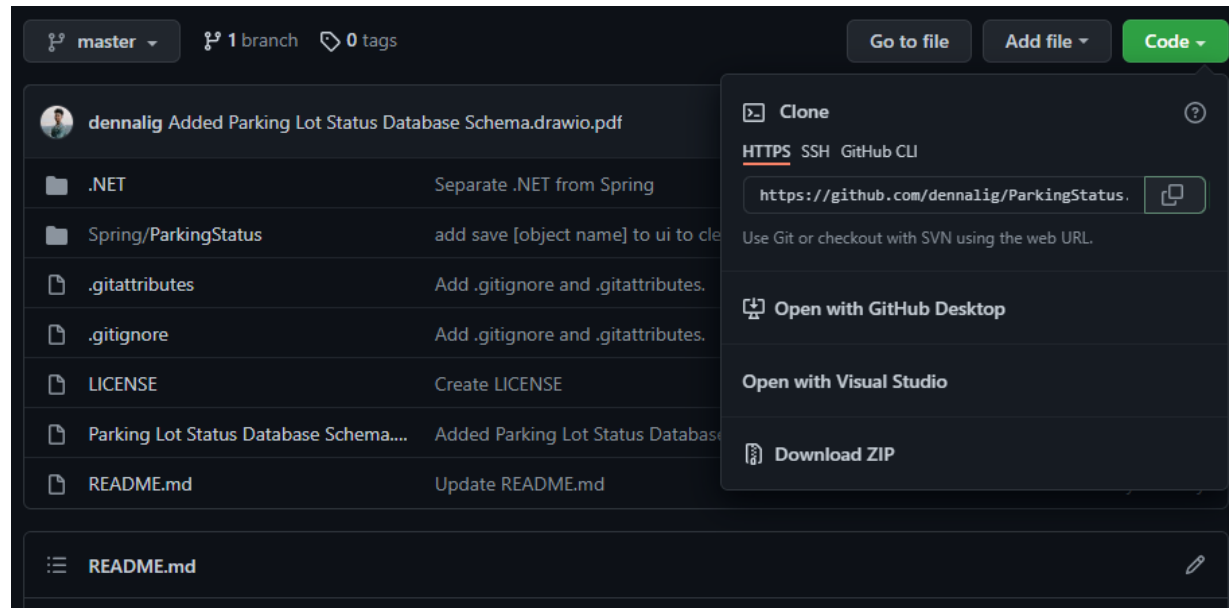
## Prerequisites

Be sure that the following are installed on your machine:

- [Git](#)
  - Run '**git --version**' to verify that it is installed.
- [Java JDK 11.0.12](#) (Parking Status was developed with Java JDK 11.0.12 and has not been tested with other versions.)
  - Run '**java -version**' to verify that it is installed.
- [Node.js](#) (Parking Status was developed with v14.17.6 and has not been tested with later versions.)
  - Run '**node --version**' to verify that it is installed.

## Cloning From GitHub

- We will first clone the GitHub repository into a local directory. Select a desired local directory or create a new one. For this demonstration, I will make a directory called "**ParkingStatus-setup**".
  - ```
mkdir ParkingStatus-setup
```
- Change into the directory
  - ```
cd ParkingStatus-setup
```
- Clone the ParkingStatus Git repository from GitHub.
  - Go to <https://github.com/dennalig/ParkingStatus>
  - Click the green '**Code**' dropdown button and copy the '**https**' link given.



- Run `git clone https://github.com/dennalig/ParkingStatus.git` in the command line:

```
ParkingStatus-setup>git clone https://github.com/dennalig/ParkingStatus.git
```

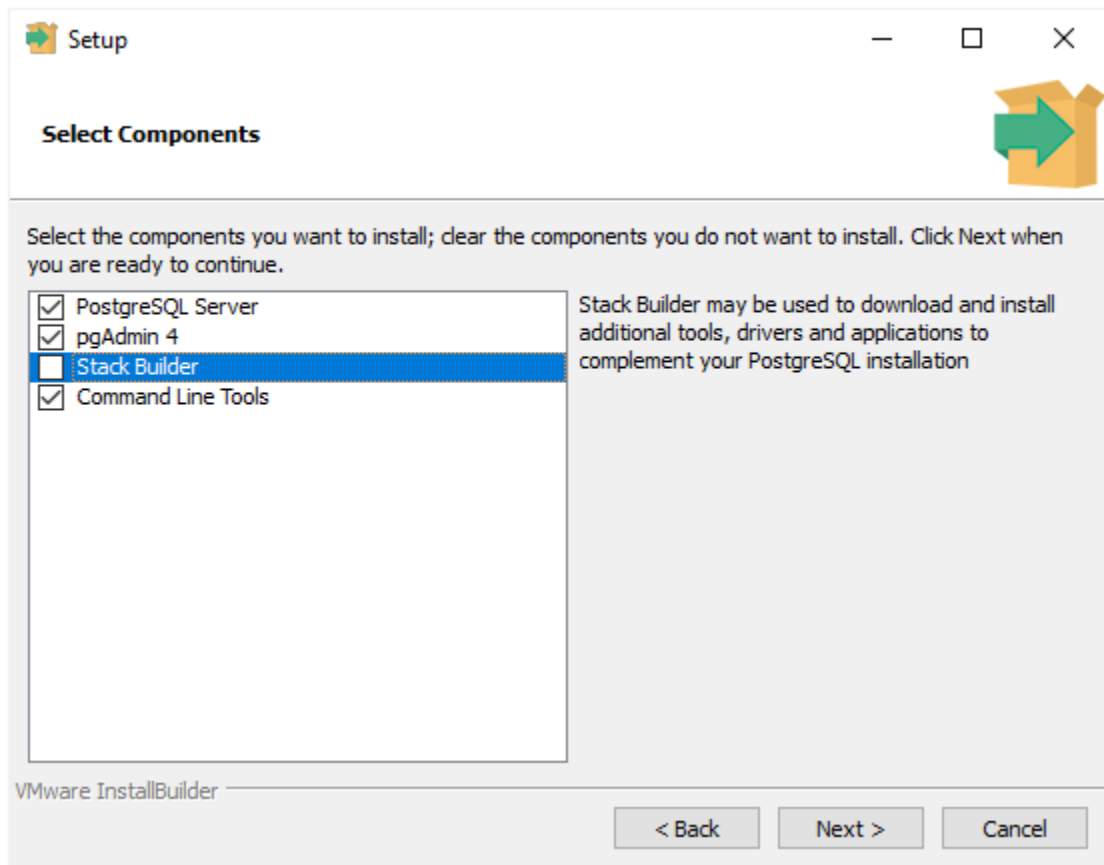
- The repository is now cloned into **'ParkingStatus-setup'**

```
C:\Users\Dennali Grissom\ParkingStatus-setup>git clone https://github.com/dennalig/ParkingStatus.git
Cloning into 'ParkingStatus'...
remote: Enumerating objects: 3261, done.
remote: Counting objects: 100% (3261/3261), done.
remote: Compressing objects: 100% (1697/1697), done.
remote: Total 3261 (delta 1701), reused 2679 (delta 1137), pack-reused 0
Receiving objects: 100% (3261/3261), 1.30 MiB | 3.09 MiB/s, done.
Resolving deltas: 100% (1701/1701), done.
```

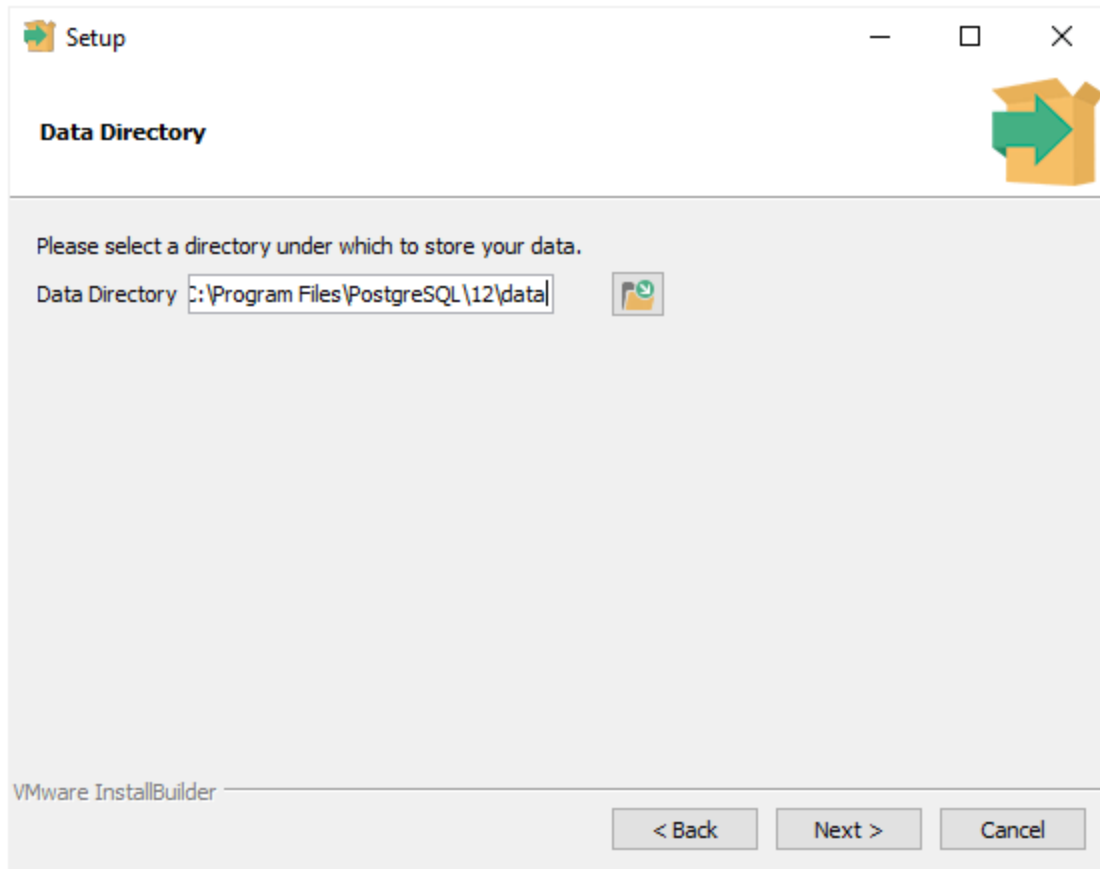
## Configuring the Database

- Configuring the database will require us to setup the **PostgreSQL** server on our machine with a few specific credentials that are recognized by backend of the application
- Go to <https://www.postgresql.org/download/> and select your operating system family
  - I will be using Windows.
- Click **'Download the Installer'** and it will take you to a list of PostgreSQL options.
  - I created the database scripts in **PostgreSQL 14**
- Select version **14.1** if on Windows.
- Run the **'exe'** file that it gives you once it is installed.

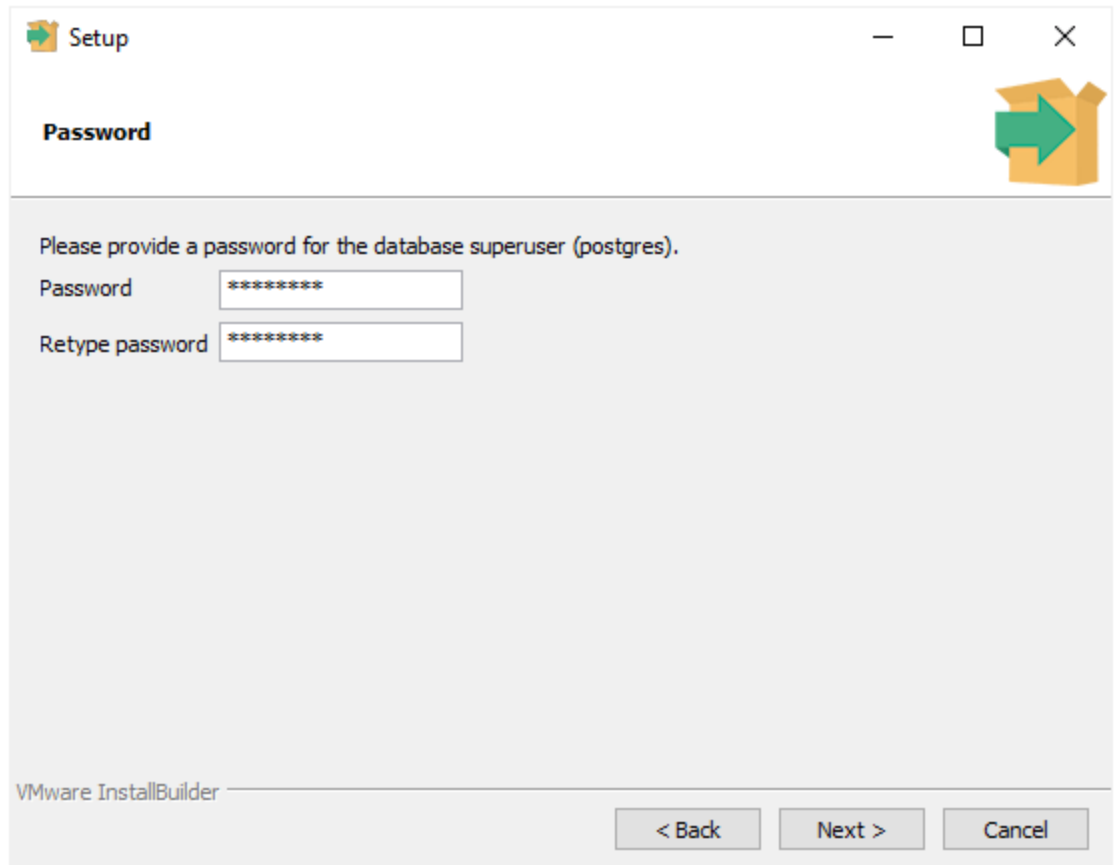
- Uncheck 'Stack Builder'



- 
- Select the default directory



- 
- We will now set up a **superuser** named '**postgres**'. These credentials are sent to the **backend** via **Spring Boot's** '**application.properties**' file and must be strictly followed.
- For the Password we will enter the Password and Retype Password as :
  - **postgrespw**
  - **(All lowercase characters)**



The screenshot shows a window titled "Setup" with a standard Windows title bar (minimize, maximize, close buttons). In the top right corner, there is a large orange arrow icon pointing right. Below the title bar, the word "Password" is displayed in bold. The main area of the window contains the text "Please provide a password for the database superuser (postgres)." followed by two input fields. The first field is labeled "Password" and the second is labeled "Retype password". Both fields contain a series of asterisks (\*\*\*\*\*). At the bottom left, the text "VMware InstallBuilder" is visible. At the bottom right, there are three buttons: "< Back", "Next >", and "Cancel".

Setup

**Password**

Please provide a password for the database superuser (postgres).

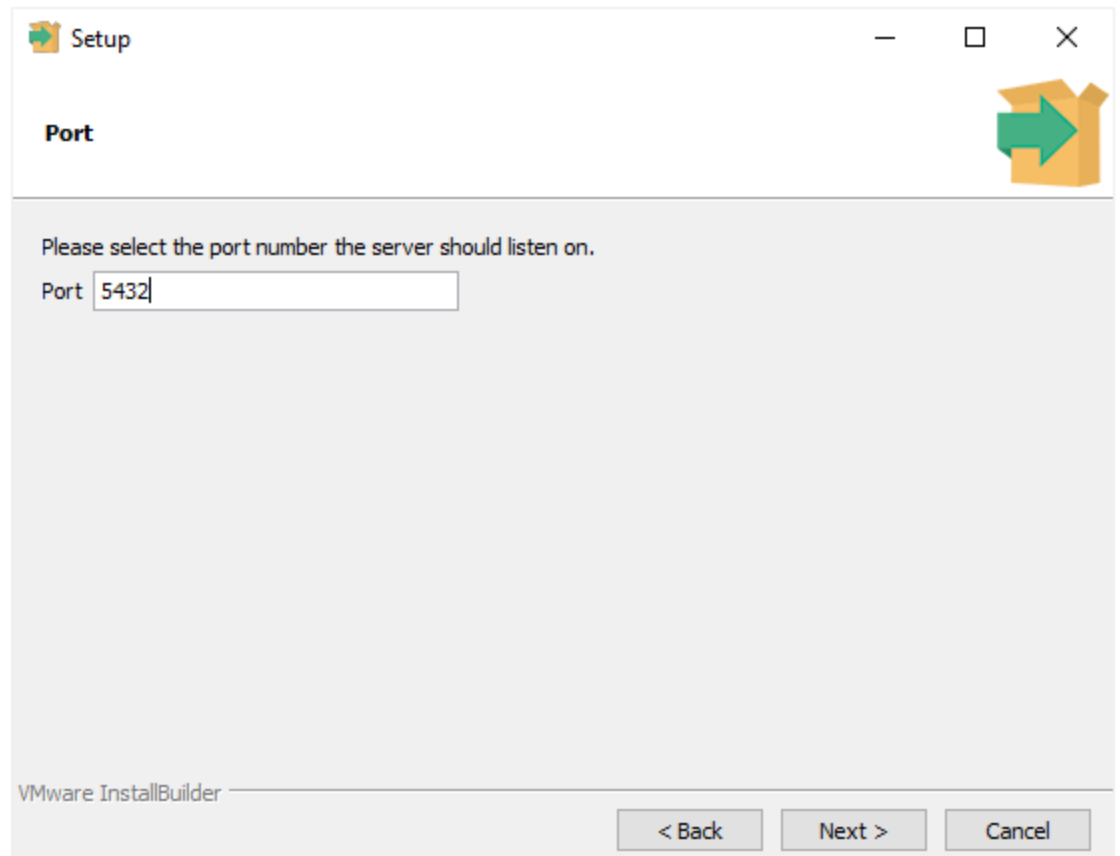
Password \*\*\*\*\*

Retype password \*\*\*\*\*

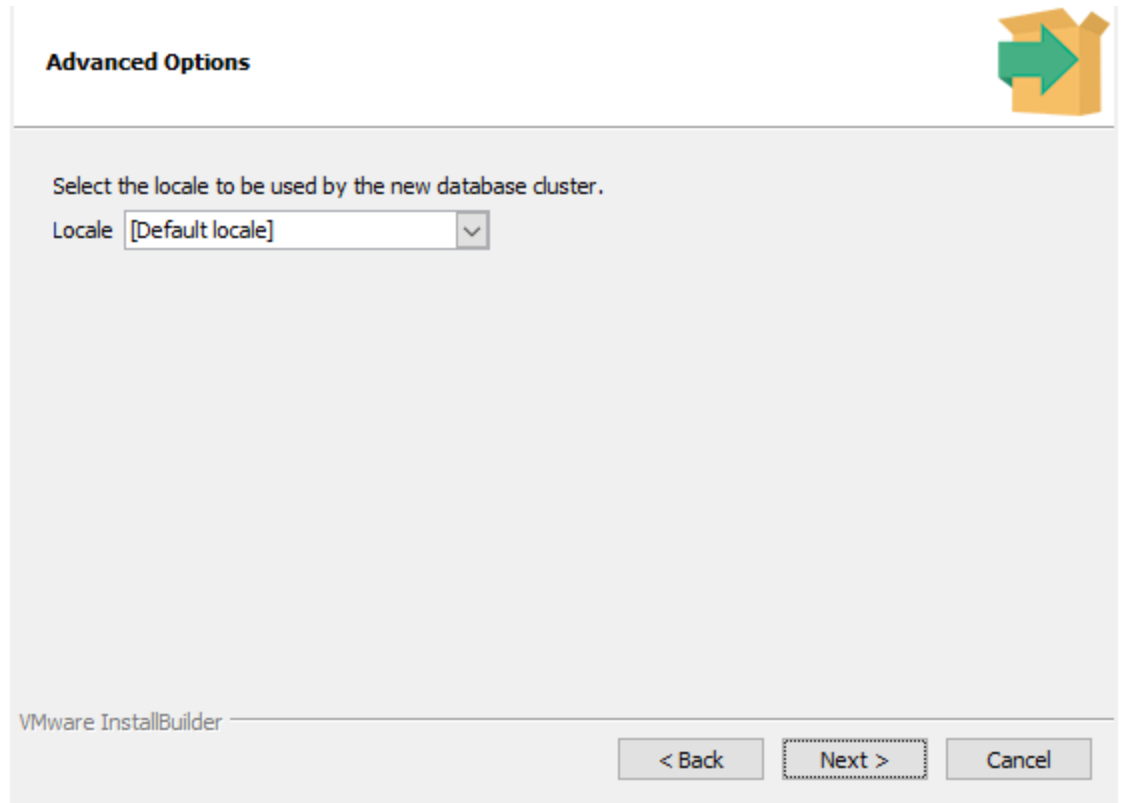
VMware InstallBuilder

< Back Next > Cancel

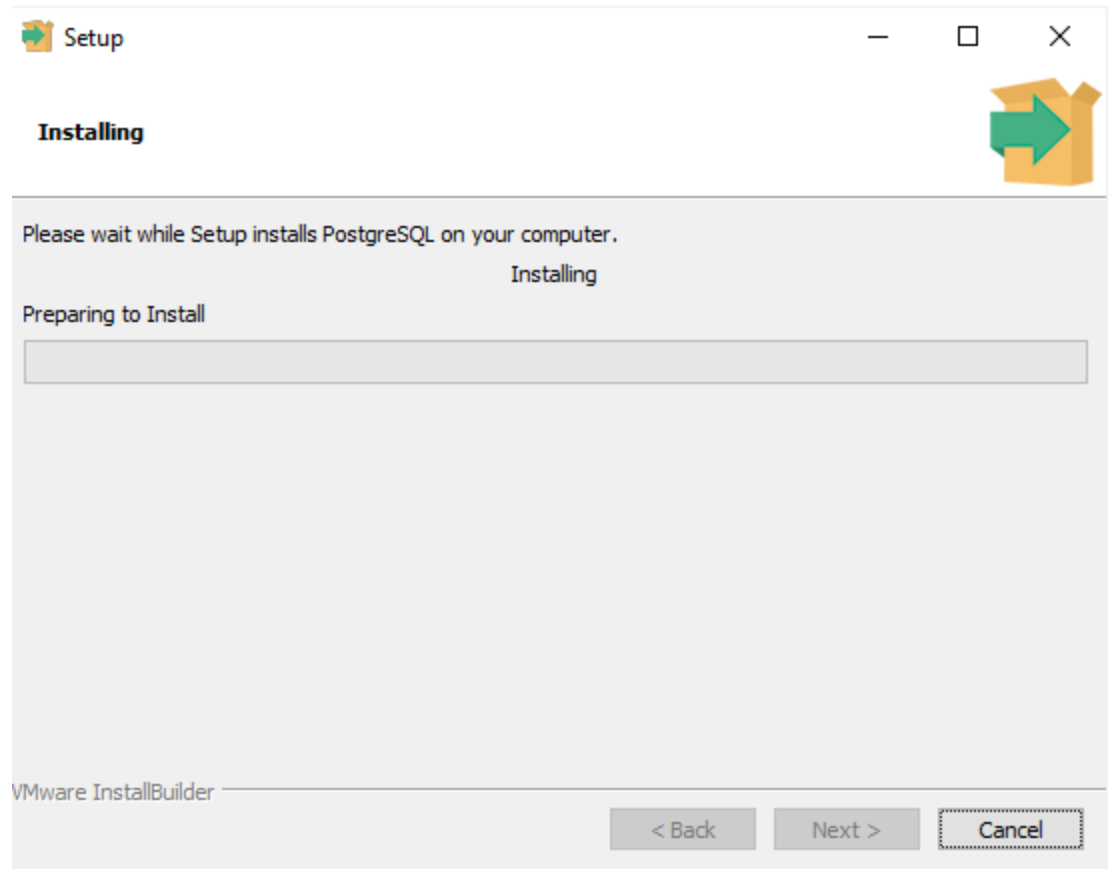
- 
- Click 'Next'
- Leave the Port number as :
  - **5432**



- 
- Leave the locale as:
  - **[Default locale]**



- 
- For the next few sections, Simply click '**Next**' until you see:

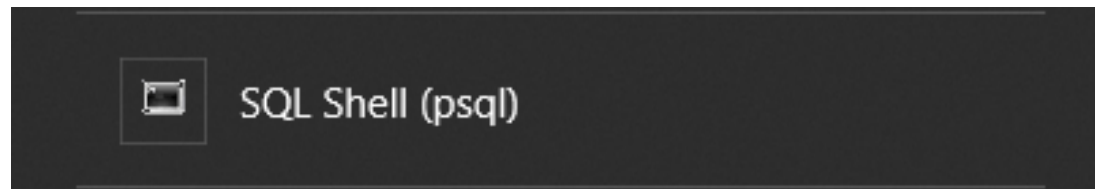
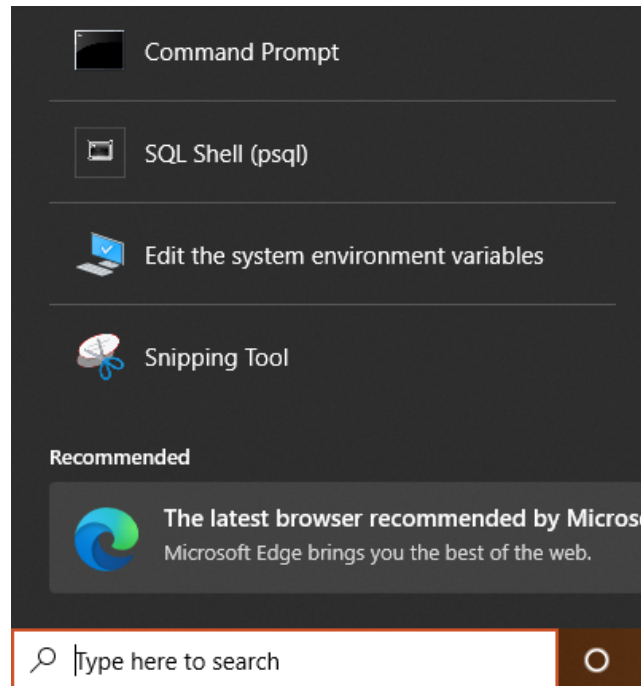


- 
- Once the installation is complete, click '**Finish**'





- - PostgreSQL server should be installed on your machine. For a more organized setup explanation of installing PostgreSQL, visit [Postgresql Tutorial](#).
  - Now we will run the PostgreSQL server to create the Parking Status **database** and specific tables.
  - In your explorer (such as Windows explorer), search 'sql' and the program 'SQL Shell (psql)' should be suggested. Click that:



- You will be presented with the following terminal:



- Press '**Enter**' on the keyboard

- Database [postgres]:
- Press 'Enter' again
- Port [5432]:
- Press 'Enter' again
- Username [postgres]:
- Enter the specified password ('**postgrespw**' (all lowercase) that we specified. NOTE: typing the password will not display the output onto the terminal. The password attempt remains hidden and is not visible.
- Press 'Enter' once the password is typed:
- Password for user postgres:
- You should now see:

```
psql (14.0)
WARNING: Console code page (437) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.
Type "help" for help.

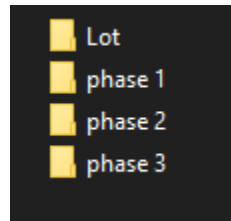
postgres=#
```

- 
- We are now connected to the server and are able to create databases.
- Run '**CREATE DATABASE parkingstatusdatabase;**' into the terminal. You should see the following message stating that the creation was successful.
  - CREATE DATABASE
- Type '**\c parkingstatusdatabase**' to connect to the newly created database named '**parkingstatusdatabase**'
  - postgres=# \c parkingstatusdatabase
  - You should see:
 

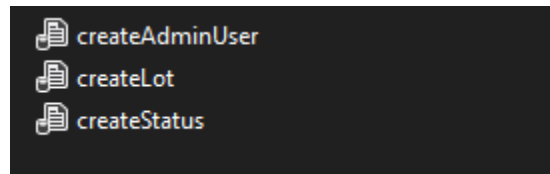
```
You are now connected to database "parkingstatusdatabase" as user "postgres".
```
- We will now create **seven** (7) tables via **SQL** scripts in '**phases**'. The reason we are creating these tables in phases is because certain tables are dependent on other ones, so the scripts need to be ran in a certain order to be successful:
  - The tables we will be creating are:
    - LOT**
    - STATUS**
    - ADMINUSER**

- iv. **LOTSTATUSCHEDULE**
- v. **LOTSTATUSCHEDULEDATE**
- vi. **STATUSEVENT**
- vii. **STATUSEVENTDATE**

- In the file explorer, we will navigate to the directory :  
**ParkingStatus-setup\ParkingStatus\Spring\ParkingStatus\src\main\resources\Data**
  - Here, you will see:



- i.
    - We will only be dealing with directories '**phase 1-3**'.
- We will copy the **SQL scripts** from here into the **PostgreSQL terminal**.
- Start with the '**phase 1**' directory and repeat the following steps for the '**phase 2**' directory and then the '**phase 3**' directory. **Run all scripts in phase 1 first, then phase 2, and then finally phase 3 (In that order).**
  - Open the directory
  - Each script will be titled as '**create**' and **then the name of the table followed by '.sql'**.



- i.
    - For each file in the directory:
      - i. **Open** the file , **copy the script**, **paste** it into the **PostgreSQL terminal**, and press '**Enter**'.
      - After Pressing '**Enter**' for each script, you should see:
        - i. **CREATE TABLE**
        - ii. This indicates that the creation of the table was successful.
- Once the previous steps were repeated for both '**phase 2 and 3**' directories, type '**\dt**' and press '**Enter**' to see a list of all tables created. You should see:

List of relations			
Schema	Name	Type	Owner
public	adminuser	table	postgres
public	lot	table	postgres
public	lotstatusschedule	table	postgres
public	lotstatusscheduledate	table	postgres
public	status	table	postgres
public	statusevent	table	postgres
public	statuseventdate	table	postgres
(7 rows)			

- If this is what you see (adminuser, lot, lotstatusschedule, lotstatusscheduledate, status, statusevent, and statuseventdate), then *congratulations*, you have successfully configured the **Parking Status Database**
- Once this is configured, you do not need to have the PostgreSQL terminal open in order for the Parking Status application to access the database.

## Configuring the Backend

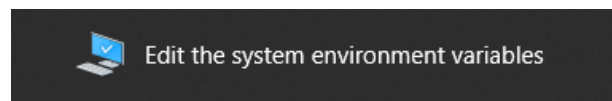
Here, we will be following a series of steps to get the **Spring Boot** backend of Parking Status configured and running.

### JAVA\_HOME Environment Variable

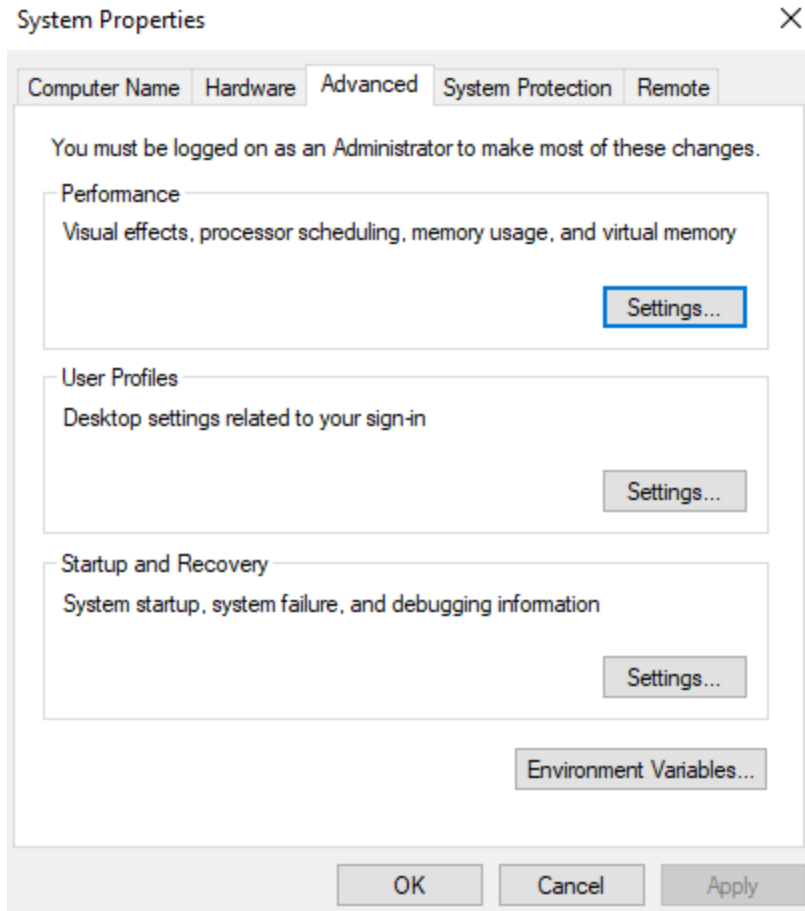
- Firstly, make sure that your **JAVA\_HOME** variable is declared to match the **location** of your **Java installation**. Otherwise, when we run ‘**mvn install**’ later, it will not work

```
Error: JAVA_HOME not found in your environment.
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation.
```

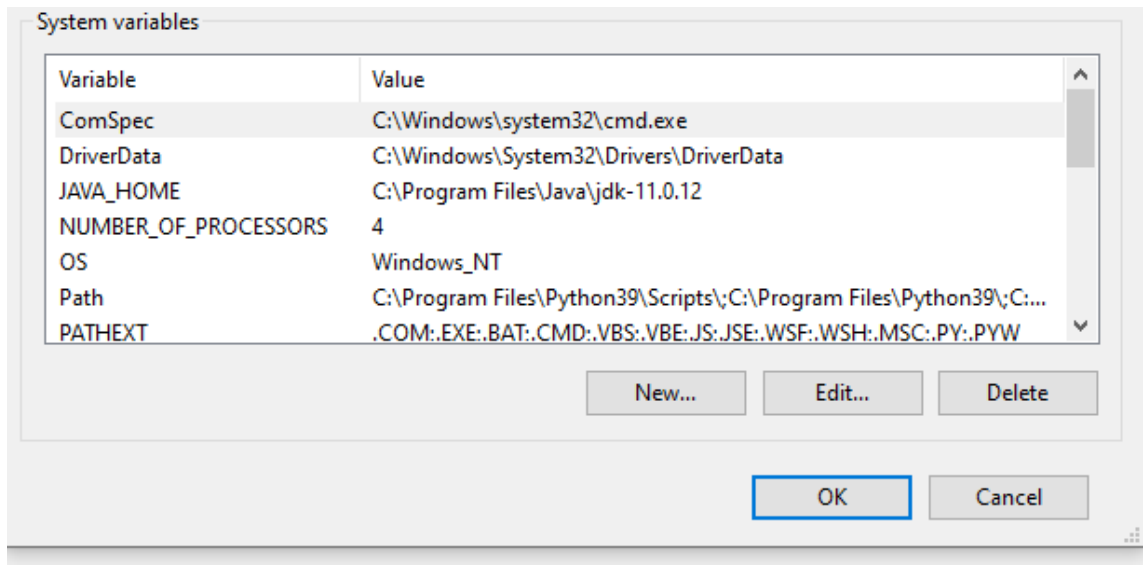
- To do this: we will go to our explorer and enter ‘**Edit the system environment variables**’ and select that option when it is suggested:



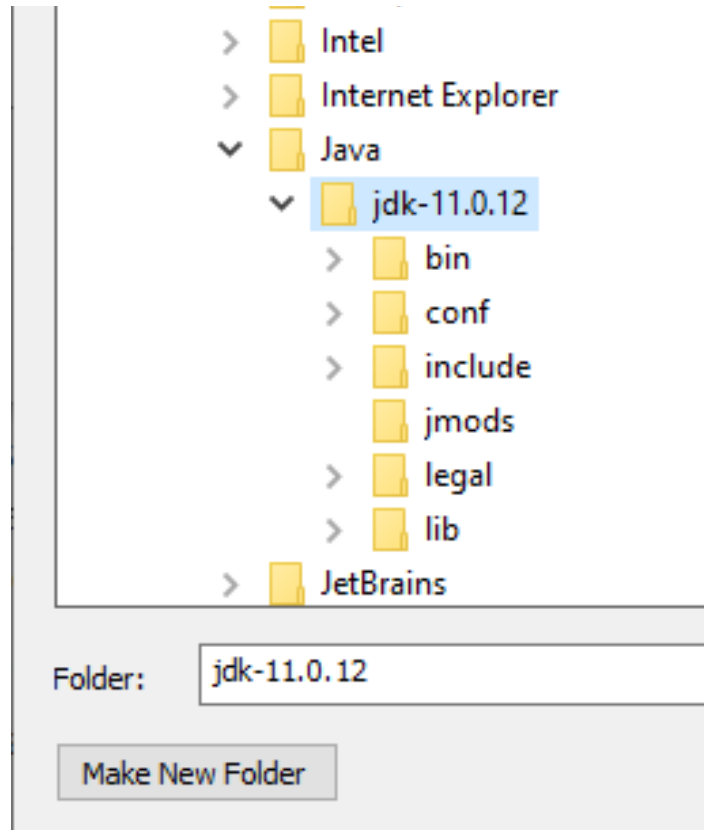
- Select ‘**Environment Variables...**’



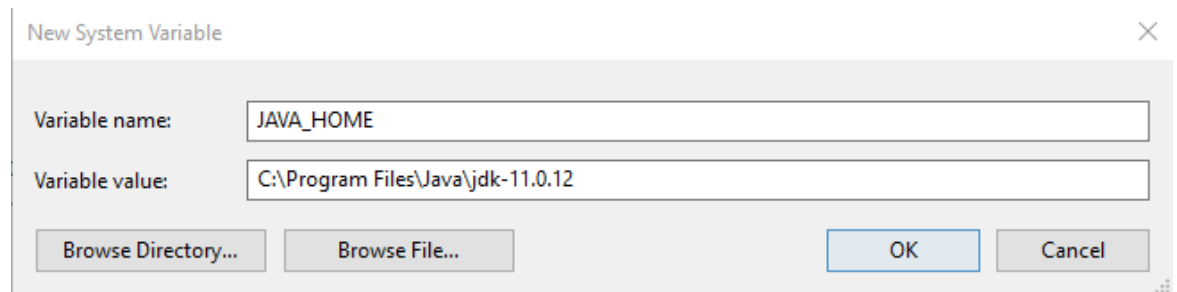
- In 'System variables' select 'New...' (NOT 'User variables for User')



- For 'Variable name:' enter 'JAVA\_HOME'
- For 'Variable value:' we will select 'Browse Directory...' and navigate to the installed Java Jdk (Typically located in: This PC > Local Disk > Program Files > Java)
- We will select the 'jdk-11.0.12' directory.



- 
- Click 'OK'
- You should see the following:



- 
- Click 'OK' again on the environment variables popup, and then 'Apply' if necessary, then 'OK'.
- Now, we should have a 'JAVA\_HOME' environment variable stored with our JDK.
- NOTE: Be sure to close out of the desired terminal you want to use to run the backend, as a new terminal will register the new changes.

## Running Maven

- We will now configure the **Spring boot backend**.
- In the a command prompt, navigate to the directory:  
**ParkingStatus-setup\ParkingStatus\Spring\ParkingStatus**
- Run “**mvnw install**”

- `ParkingStatus-setup\ParkingStatus\Spring\ParkingStatus>mvnw install`

- This may take a while, however at the end of the process, you should see:

- ```
INFO] -----
INFO] BUILD SUCCESS
INFO] -----
INFO] Total time: 06:55 min
INFO] Finished at: 2021-12-04T19:08:08-05:00
INFO] -----
```

- This means that we have successfully generated a **.jar** file that we will run in order to get the backend of Parking Status up and running.

- We will now navigate into the ‘**target**’ directory.
  - Run ‘**cd target**’
- To run the backend we will now enter ‘**java -jar ParkingStatus-0.0.1-SNAPSHOT.jar**’.

- `ParkingStatus\target>java -jar ParkingStatus-0.0.1-SNAPSHOT.jar`

- You should see a ‘Spring’ logo and then a large amount of text
- The last line of text should contain the words ‘**Started ParkingStatusApplication**’



```
Tomcat started on port(s): 8080
Started ParkingStatusApplication
```

- *Congratulations*, the backend is now running.
- To see the server running, navigate to ‘<http://localhost:8080>’ on a browser.
- To view the **API endpoints**: add ‘**api/v1/**’ and the name of your desired endpoint (‘lots’, ‘status’, ‘statusevents’, or ‘adminusers’).
- To end the Server press ‘**Ctrl-c**’.
- NOTE: In order for the data to be displayed on the frontend, the Spring Boot server must be running simultaneously with the frontend.










## Configuring the Frontend

Don't worry, we are almost done...


- Here we will configure and run the **frontend of ParkingStatus** using **Node.js**.
- Open up another **Command Prompt** and **cd** into your chosen directory for Parking Status again.
- Run '**cd ParkingStatus\Spring\ParkingStatus\src\js\parkingstatus.web**'

- `\ParkingStatus-setup\ParkingStatus\Spring\ParkingStatus\src\js\parkingstatus.web>`

- In the Windows file explorer, go to this same '**parkingstatus.web**' directory

| Name                                                                                             | Date modified      | Type               | Size   |
|--------------------------------------------------------------------------------------------------|--------------------|--------------------|--------|
|  public         | 12/4/2021 11:40 PM | File folder        |        |
|  src            | 12/4/2021 11:40 PM | File folder        |        |
|  .gitignore     | 12/4/2021 11:40 PM | Text Document      | 1 KB   |
|  package        | 12/4/2021 11:40 PM | JSON Source File   | 2 KB   |
|  package-lock | 12/4/2021 11:40 PM | JSON Source File   | 708 KB |
|  README       | 12/4/2021 11:40 PM | Markdown Source... | 4 KB   |
|  tsconfig     | 12/4/2021 11:40 PM | JSON Source File   | 1 KB   |

- Delete the '**package-lock.json**' file.

- |                                                                                                  |                    |                  |        |
|--------------------------------------------------------------------------------------------------|--------------------|------------------|--------|
|  package-lock | 12/4/2021 11:40 PM | JSON Source File | 708 KB |
|--------------------------------------------------------------------------------------------------|--------------------|------------------|--------|

- The '**parkingstatus.web**' should not have a package-lock.json file now

- In the command prompt, run '**npm install**'

- `\src\js\parkingstatus.web>npm install`

- You should see npm packages being installed (this may take awhile):

```
npm WARN deprecated babel-eslint@10.1.0: babel-eslint is now @babel/eslint-parser.
[.....] / fetchMetadata: sill pacote version manifest for detect-port-
```

- Once all of the npm packages are installed, you should see a summary of the installation

```
added 1972 packages from 858 contributors and audited 1977 packages in 269.404s

157 packages are looking for funding
  run `npm fund` for details

found 13 vulnerabilities (8 moderate, 4 high, 1 critical)
  run `npm audit fix` to fix them, or `npm audit` for details

C:\Users\Dennali Grissom\ParkingStatus-setup\ParkingStatus\Spring\ParkingStatus\src\js\parkingstatus.web>
```

- Do not mind the vulnerabilities listed here as these are not malicious files.
- Now the Parking Status frontend is installed.

- Run '**npm start**' to run the frontend React application.

- ```
\src\js\parkingstatus.web>npm start
```

- After the development server has been started, you should see

```
Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

○

- *Congratulations*, you have successfully configured the frontend. To view the frontend of the application, navigate to '<http://localhost:3000>' (it should be automatically opened once the development server is run).



(Timezone:EST)

North lot	+
Test parking lot	+
Sanderson Lot	+
Probasco Lot	+
Andreas Lot	+
Shadowlands Lot	+

- 
- To end the server, Run '**Ctrl-c**' and answer '**Y**' to '**Terminate batch job (Y/N)?**'.

Congratulations, Parking Status is fully up and running on your machine! Go make peoples' lives easier.

## Things To Note

- You do not need to have the PostgreSQL terminal up and running or be connected to the parkingstatusdatabase via the terminal in order to alter data in it via the UI.

- You *do* need to have both the Spring Boot backend and React frontend running at the same time to interact with the data. Spring Boot should always be running on '<http://localhost:8080>' and React should always be running on '<http://localhost:3000>'. The easiest way to do this is to open a new command prompt for both parts of the application.
- In order to **alter** Parking Status's data in any way, you must sign up as an **admin user**. This is done by going to the frontend, clicking '**Sign Up**', entering an **email address**, entering a **password** (of at least one character long), and clicking **Submit**. If there are no error messages on the UI, navigate to '**Admin Login**' and re-enter your submitted **email** and **password** and click '**Login**'. If the login is successful, your email address should appear on the top left corner of the navigation bar, above the 'Parking Status' title.

Enjoy!

