**How to Use Environment Variables Like a Pro in Node.js -** [Bewar Salah](https://bewar-usman.medium.com/?source=post_page-----7fd2793808d3--------------------------------) Oct 27, 2021

**Introduction**

There are many constant values that need to be stored in applications and retrieved for later use, like DB username, DB password, email password, and API Keys. These values may change in different circumstances. When a developer writes a piece of code, the code needs to be tested. The tests should be against the development environment. The developer should not connect to a production DB server.

The developer usually uses a local database for developing and testing, a personal email account to send test emails. Sometimes, before production deployment, the app goes to the staging environment and is tested by the Quality Assurance (QA) team against the staging environment. Finally, in the production environment, all services should be ready for production. For that reason, we need to maintain two or more environments while creating applications.

It is convenient and conventional to store some constant values which differ during the environment change. The application can be configured to retrieve these values based on the environment it is running on.

**Environment Variables**

Environment variables can be created on the application or system level.

Chart

Description automatically generated

Different Levels

Operating systems like Windows and Linux manage their own environment variables. Node.js application can read these values from the hosting system.

But wait! What if the developer wants to read these values locally stored in a file instead (for example, a configuration file)? Node.js applications can also manage their own environment variables stored in a file (called configuration file in most other programming languages). Node.js cannot maintain the local environment variables by itself. It has to use a package called dotenv.

A picture containing text

Description automatically generated

dotenv reads from .env

The package dotenv can be installed using an NPM command and it reads from a file called .env .

**Set Up an Application**

To understand the whole concept, we are going to create a Node.js server running on multiple environments to demonstrate the usage of dotenv node package for maintaining local variables, and also try to deploy on a server to read system variables while running on the server.

Steps for setting up the application:

* Creating package.json file

npm init -y

* Installing dotenv
* npm i -S dotenv
* Calling dotenv config function on the top of the application

require("dotenv").config();

* Creating .env file in the root of the application with the following content

Shape

Description automatically generated with medium confidence

.env file

* Accessing the values using process.env

const DB\_HOST = process.env.DB\_HOST;  
const DB\_USER = process.env.DB\_USER;

*Now, if we run the application locally, dotenv reads these values from****.env****file and attached to process.env object. If****.env****does not exist, node.js tries to read these values from the machine environment variables.*

The below application reads the values from .env and return it to the user.

Text

Description automatically generated

complete node.js server

**Deploying on Heroku**

We are going to deploy the application on **Heroku**, and explain how to read these values from **system variables**.

There are multiple ways to deploy applications on **Heroku**. We are going to use the **GitHub** method.

**Requirements**

* Github account
* Heroku account
* Git installed on a working machine

**Steps to deploy**

* 1. Create a GitHub account and create a new git repository.

Graphical user interface, text, application, chat or text message

Description automatically generated

New Repository

The name of the repository should be filled in our case, it is **env\_demo**. If the project is private, you need to select **Private** option.

Graphical user interface, text, application, email

Description automatically generated

github — new repository

2. We need to make some changes in **package.json.**

* add engines and specify node version — this is required for **Heroku** to install the node for the app.
* add start script node index, **Heroku**runs npm start for Node.js apps.

{  
 *"name"*: "env\_demo",  
 *"version"*: "1.0.0",  
 *"description"*: "",  
 *"main"*: "index.js",  
 *"scripts"*: {  
 ***"start"*: "node index",**  
 *"test"*: "echo \"Error: no test specified\" && exit 1"  
 },  
***"engines"*: {  
 *"node"*: "14.x"  
 },**  
 *"keywords"*: [],  
 *"author"*: "",  
 *"license"*: "ISC",  
 *"dependencies"*: {  
 *"dotenv"*: "^10.0.0"  
 }  
}

Also, we need to specify the port to run the app in **index.js**file as below. This is required by **Heroku**. You should not specify the port.

http  
 .createServer(*function* (*req*, *res*) {  
 *res*.writeHead(200, { "Content-Type": "application/json" });  
 *res*.write(JSON.stringify(ENV\_VARS));  
 *res*.end();  
 })  
 .listen(**process.env.PORT**);

3. Add a file named **.gitignore**, in the root directory of the project. We can specify the files that should not be uploaded to the **GitHub** server. In our instance, we don’t want to add **.env**, as it contains my personal details and **node\_modules** directory as it is not recommended, **Heroku**creates **node\_modules** and fetches the required **npm**packages.

Graphical user interface, text, application, Teams

Description automatically generated

.gitignore file

4. In the project directory on your machine, write the below commands.

git init  
git add .  
git commit -m "initial commit"  
git remote add origin <https://github.com/bewarusman/env_demo.git>  
git push origin master

5. Go to **Heroku** and create an account

6. Create a new app

Graphical user interface, text, application

Description automatically generated

heroku — create new app

7. Choose **Connect to Github** from the **deploy** tab.

Graphical user interface, application, Teams

Description automatically generated

heroku — connect to github

8. Find the repository and choose to **Connect**, then click **Deploy Branch**. After the deployment is completed, click on the **View** button, to see the app in the browser.

Graphical user interface, text, application, email

Description automatically generated

heroku — connect

Graphical user interface, text, application

Description automatically generated

heroku — deploy branch

Now the app opens and displays an empty **JSON** object. Why? because we did not specify the environment variables in the **Heroku** ecosystem.

Text

Description automatically generated with medium confidence

node — reading env variables

**Configuring Variables**

* In the settings tab, click on **Reveal Config Vars**.

Graphical user interface, text, application, email

Description automatically generated

* Add the variables there.

A screenshot of a computer

Description automatically generated

heroku — config vars

* and testing...

Graphical user interface, text, application

Description automatically generated

**Conclusion**

It is very crucial in application development to separate multiple environment variables. Maintaining multiple environment variables is easy in a Node.js application with the usage of the npm package dotenv. It reads from a file called .env. While a **.env** file does not exist, **process.env** tries to read from the machine environment variables.