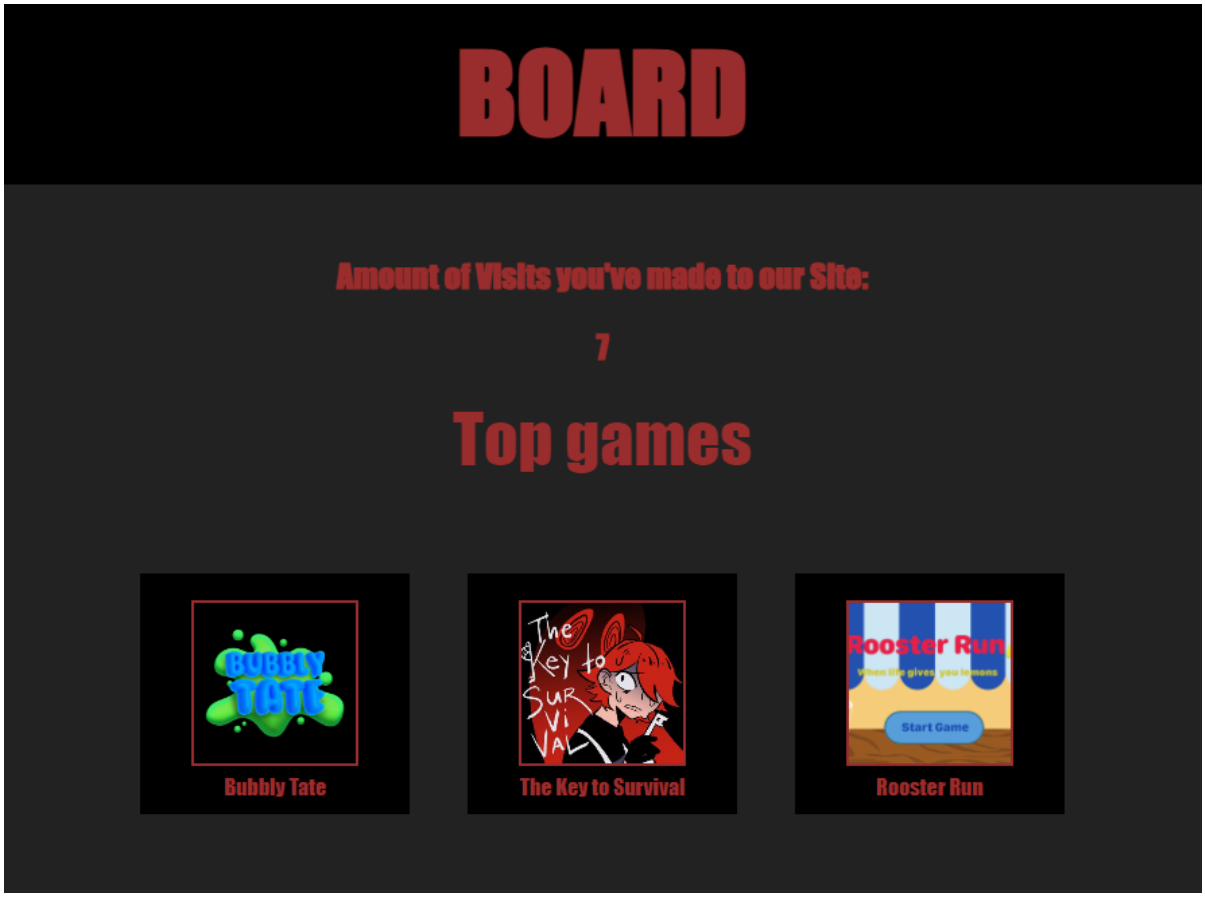
Unix LIA Report

Project Description

Our project is a website hosted on a virtual private server that allows the user to download the jar files and the standalone Greenfoot files of the available games. The website shows the number of visits that it gets on the same device, and this number is updated automatically.

Link to the repository: https://github.com/yumvinegar/Site-for-Unix.git

Website: https://whale-app-qwckd.ondigitalocean.app/



Pictured above is the final version of our website; it allows the user to view a video demo of a game and to download either the jar file or the standalone greenfoot project of the game.

Our initial goal for the website was to allow the user to play the games on the site itself. To do this we tried to research how to export a greenfoot game and embed it onto an html file. We exported a greenfoot game as a jar file and tried to embed it into an html file using an applet. However, this did not work because modern browsers do not support applets; they are deprecated in HTML5.

```
game.html > html
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Rooster Run</title>
6   </head>
7   <body>
8       <APPLET ARCHIVE="RoosterRun.jar" CODE="Rooster.class" WIDTH=400 HEIGHT=200>
9       </APPLET>
10  </body>
11  </html>
```

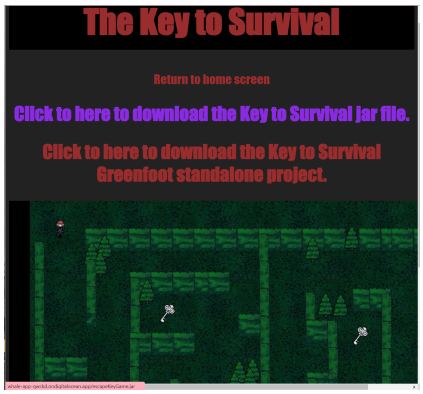
We then tried to research additional methods to export a Greenfoot game onto an html file without the use of an applet. We discovered Tea Vm which is a sort of compiler that transforms java code into Javascript.

We figured that perhaps we could try to convert the jar file into javascript and then use the javascript in our html file. However, Tea Vm no longer allows users to download their program. We tried to launch the program through a gradle project, as mentioned on their website but were unable to do so.

Our next strategy was to try to post our games on the greenfoot website itself . The Greenfoot website offers a code that allows you to embed a game if you post it on their website. So we wanted to post our games on the greenfoot website and embed them onto our own website afterwards.

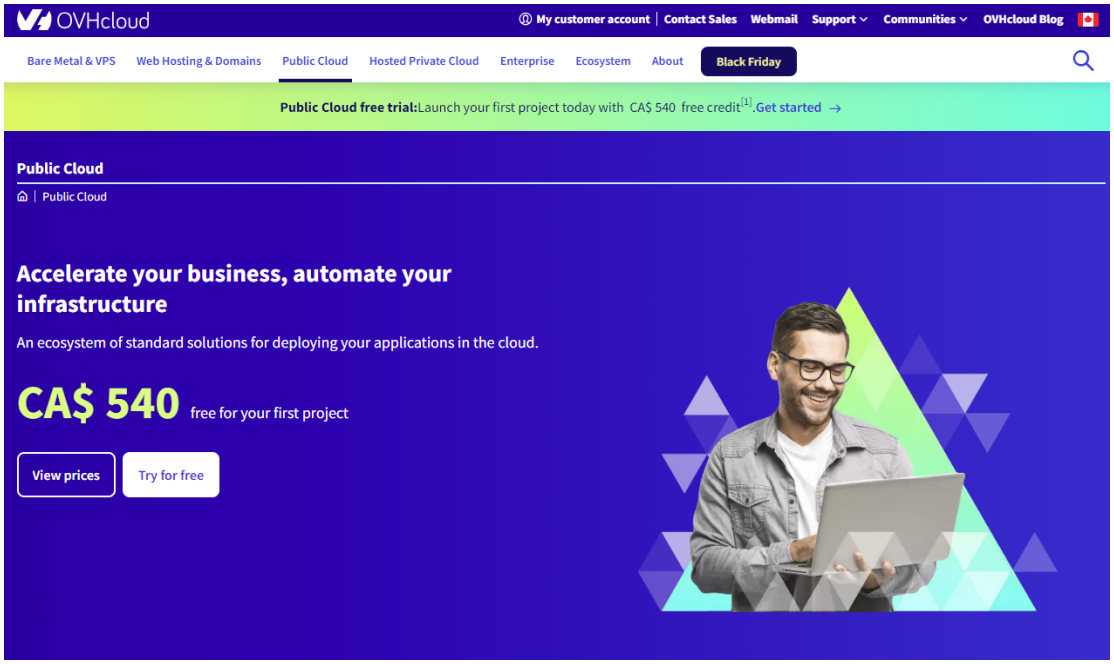
Unfortunately, when trying to export our games onto the Greenfoot website, the process was very long and it did not appear to be working. The application said that the game's resources were being bundled for over 4 hours, with no end in sight. So we decided to end the process. Instead of trying to post the game onto the greenfoot website, we decided to screen record a portion of the games being played. We then added these videos to our site. We also added links that allow the user to download the jar file for the games and the greenfoot .gfar files (standalone projects).

```
<!DOCTYPE html>
<html>
    <header>
        <link rel="stylesheet" href="style.css">
        <title>Board | Game 3</title>
    </header>
    <body>
        <main>
            <h1>Rooster Run</h1>
            <a href="index.html">
                <p id="back">Return to home screen</p>
            </a>


            <a id = "back" href="RoosterRun.jar"><h2>Click to here to download the Rooster Run jar file.</h2></a>
            <a id = "back" href="RoosterRun.gfar"><h2>Click to here to download the Rooster Run Greenfoot standalone

            <div class="main">
                <!-- this div is for game -->
                <video autoplay muted controls="controls"  loop width="1060" height="750">
                    <source src="RoosterRunDemo.mp4" type="video/mp4">
                </video>

            </div>
        </main>
    </body>
</html>
```

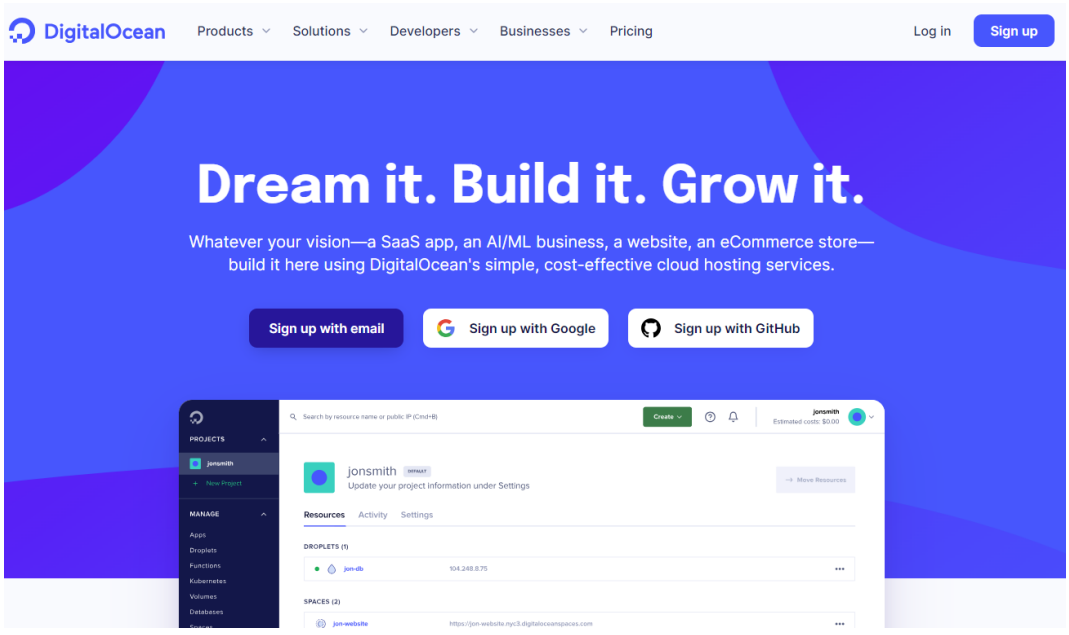


Virtual Private Server Options

When deciding on a Virtual Private Server host, we narrowed it down to two options: OVHCloud and DigitalOcean. We compared the data we gathered from the reviews of various users.



https://www.ovhcloud.com/en-ca/

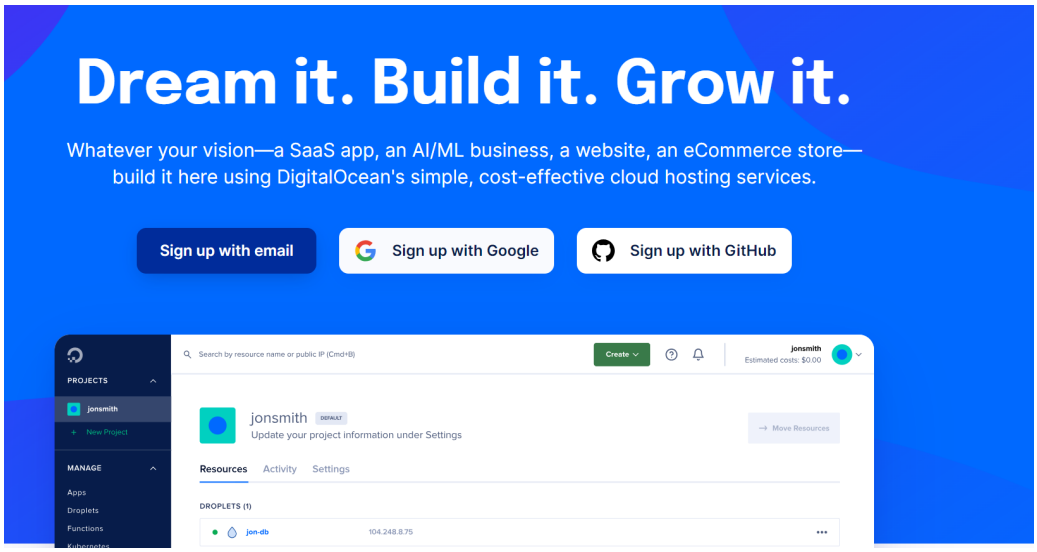


https://www.digitalocean.com/

Ultimately we decided to go with Digital Ocean to provide our VPS because it has all the features we need; it allows us to host our website. It offers a shell/console, it offers efficient CPU usage and it is available at a low cost. Digital Ocean is also a known cloud hosting provider so they are reliable.

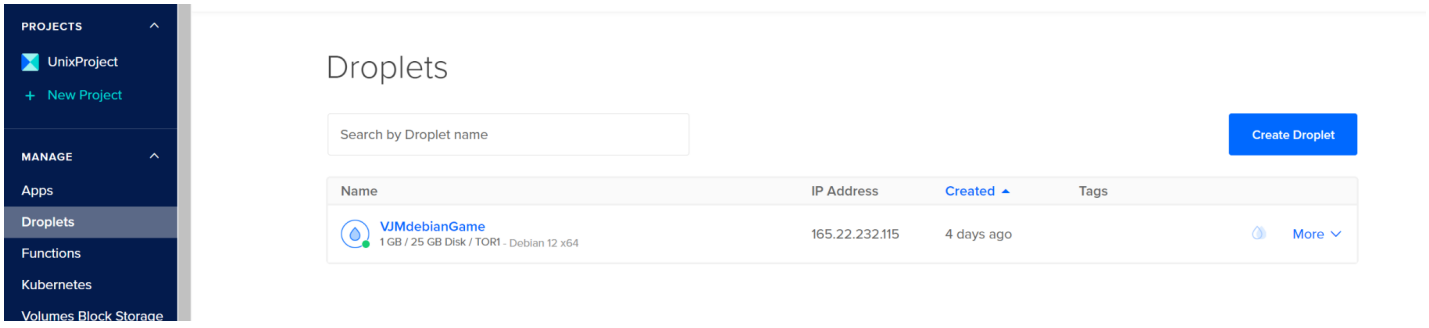
Steps to Setting up the VPS:

1. Create a Digital ocean account



The first step is to create your Digital Ocean account. You can do so by email, or by linking your gmail or github accounts. Once you select the account you wish to link to your Digital Ocean account you will be asked to enter your payment information. You will not be charged immediately, however please be advised that if you input the wrong card information your account will be locked and you will have to create a new one.

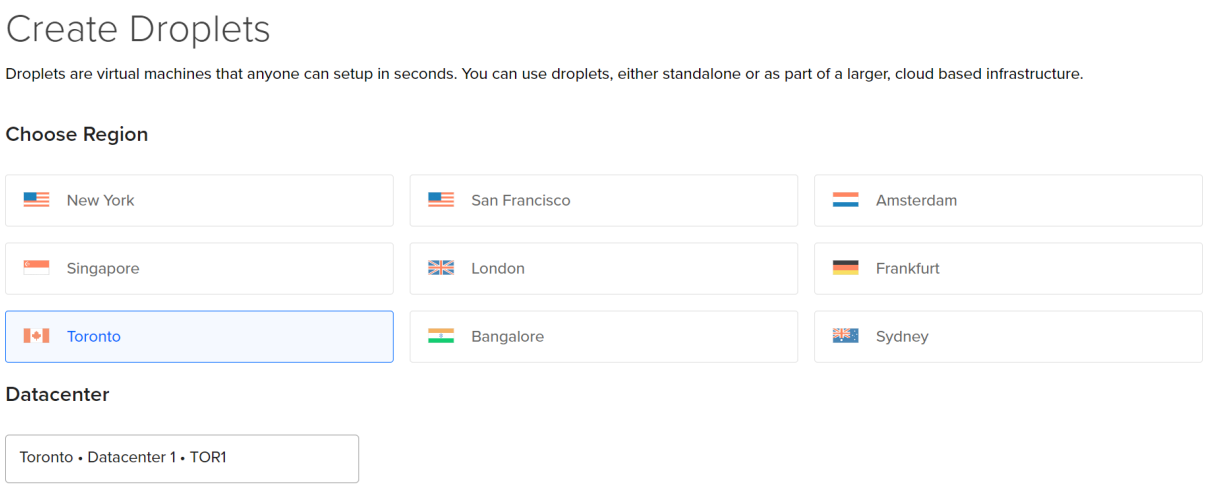
2. Create a new droplet



You can create a new droplet by clicking “Droplet” on the “Manage” menu located on the left hand side of the screen. Then click on “Create Droplet”.

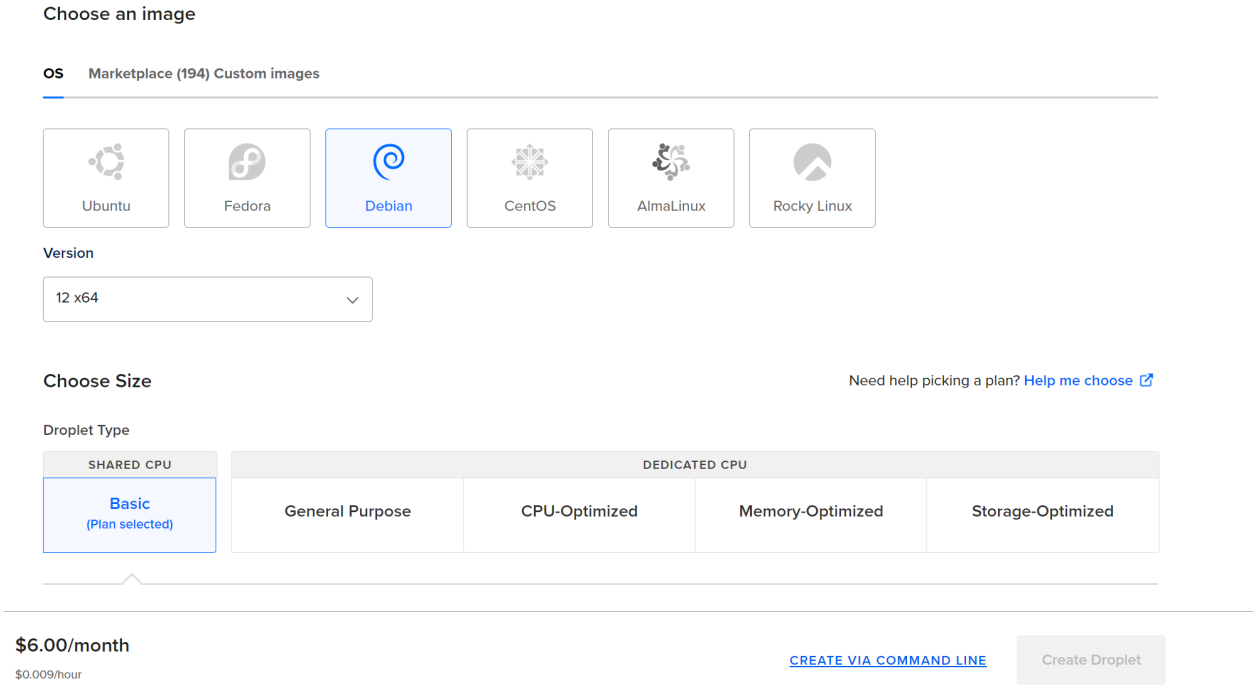
3. Select Region

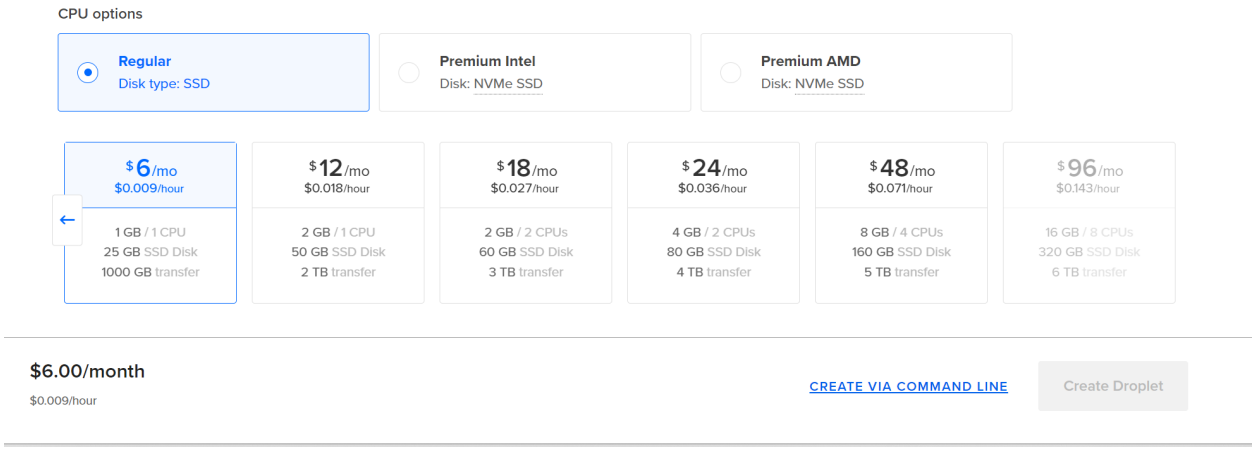
You will be asked to select your region, choose the region closest to you.



4. Select Image, Plan Type and CPU options

You will be asked to select the image and plan type for your droplet. We have chosen a debian image and the basic plan with regular CPU options for the purposes of our project as we only require basic features.

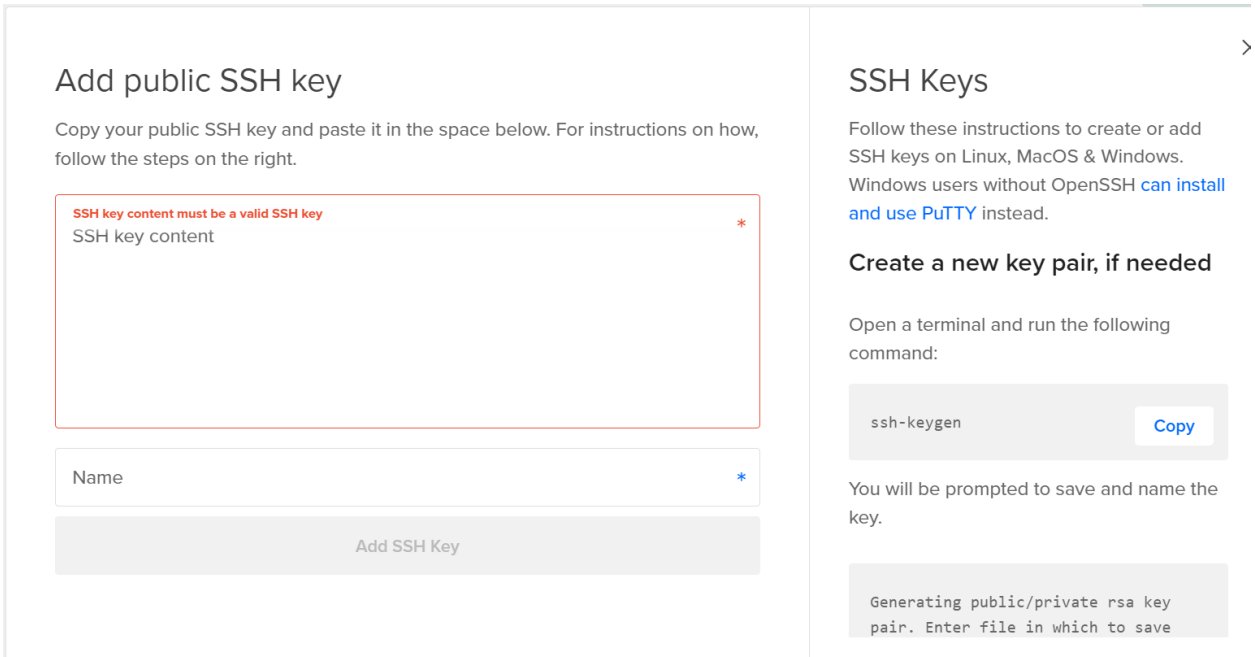




5. Authentication method -> Create SSH Keys

Next you will be asked to choose an authentication method, the two options are SSH Keys or password. Select SSH Key, this will allow more security for your server and will also allow each team member to connect to the server individually.

To create your ssh key, click on the "New SSH Key" button.



You must open a terminal and run the command : ssh-keygen. You will be asked to enter a path to save your key into, the default is (/.ssh/id_rsa), this is the recommended path to store your ssh key, I only changed the path for the purposes of this tutorial.

```
Thu Dec 07  jayda@debian:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jayda/.ssh/id_rsa): tutorialExample
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in tutorialExample
Your public key has been saved in tutorialExample.pub
The key fingerprint is:
SHA256:l0h4spQCUBFY1P96pBjW+45GpJ58LVzLSNmeMX7vT7g jayda@debian
The key's randomart image is:
+---[RSA 3072]----+
|.*B+             |
|.  ..  o         |
|    ..= o        |
|     oo= . .     |
|     +.+S o      |
|    + = B.   .   |
|   + B % =  . .  |
|    = X.O .  o   |
|     o.=o. oE..  |
+----[SHA256]-----+
```

6. Add your public key to the server

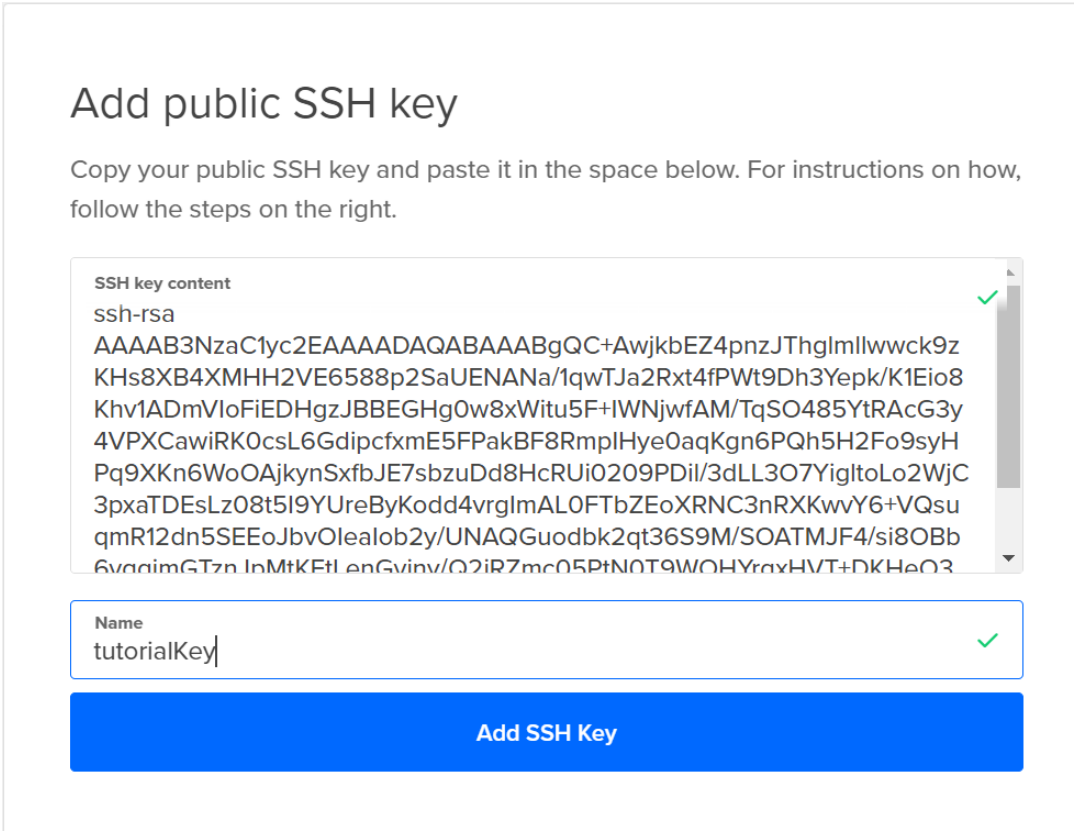
To retrieve the contents of your SSH key, run the following command:

cat ~/.ssh/id_rsa.pub

(if you have changed the path of your ssh key, use cat yourPath.pub)

```
Thu Dec 07  jayda@debian:~$ cat tutorialExample.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC+AwjkbEZ4pnzJThglmllwwck9zKHs8XB4XMHH2VE6588p2Sa
UENANa/1qwTJa2Rxt4fPWt9Dh3Yepk/K1Eio8Khv1ADmVIoFiEDHgzJBBEGHg0w8xWitu5F+IWNjwfAM/TqSO48
5YtRAcG3y4VPXCawiRK0csL6GdipcfxmE5FPakBF8RmpIHye0aqKgn6PQh5H2Fo9syHPq9XKn6WoOAjkynSxfbJ
E7sbzuDd8HcRUi0209PDil/3dLL3O7YigltoLo2WjC3pxaTDEsLz08t5I9YUreByKodd4vrgImAL0FTbZEoXRNC
3nRXKwvY6+VQsuqmR12dn5SEEoJbvOIeaIob2y/UNAQGuodbk2qt36S9M/SOATMJF4/si8OBb6vqgimGTznJpMt
KFtLenGvjny/Q2iRZmc05PtN0T9WOHYrgxHVT+DKHeO3wno+xF9RWGTLIwABh56It92jp/4SL6phSpXFWqb6kgi
bqcMDa7wkN54sP9EHaRcMW2jWK5MM= jayda@debian
```

Then copy and paste your public key (the results of this command) into the designated spot in in digital ocean (the form that pops up upon clicking add new key):

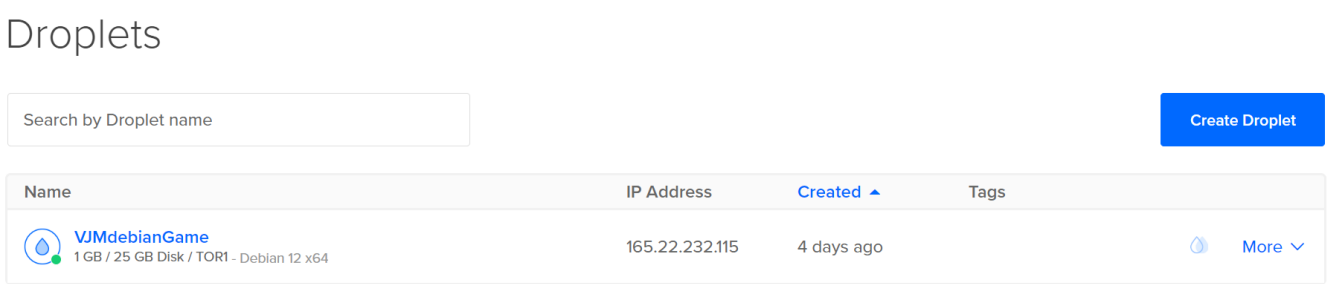


Repeat these steps for all of the team members so that everyone can connect to the server individually.

7. Choose Hostname and Create Droplet

After adding all the ssh keys, you can choose your host name and finalize your drop by pressing “Create Droplet”.

This is our droplet(VPS), it’s IP Address is 165.22.232.115



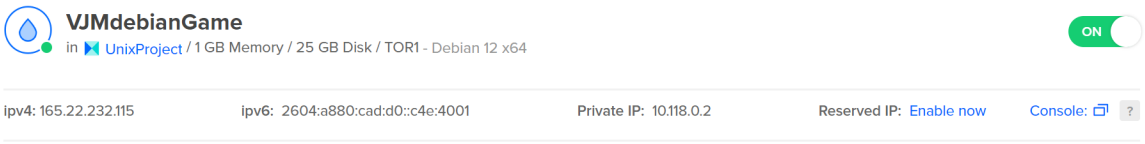
Changing the default port of the server

We decided to change the default ssh port of the server (22) since this opens the server up to being targeted by hackers. The new default port of the server is 8089.

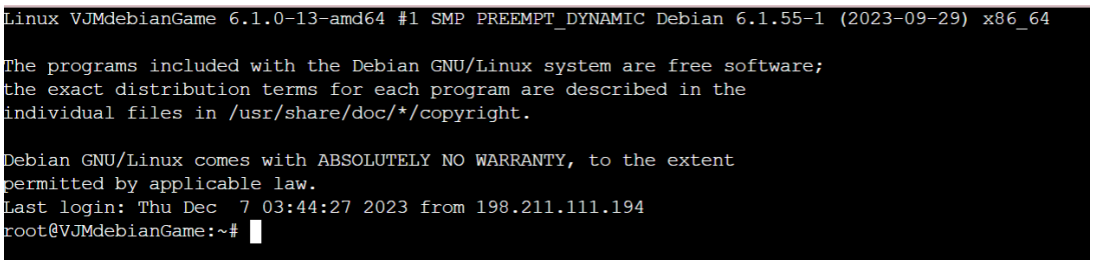
Steps for changing the default SSH Port:

1. Open up the console

If you do this through the digital ocean site you will automatically be the root user. If you are connecting via another machine, you will need to gain root privileges.

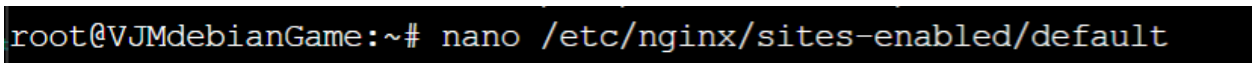


Open your droplet and click on “Console” in the top right corner. A terminal will then be opened:

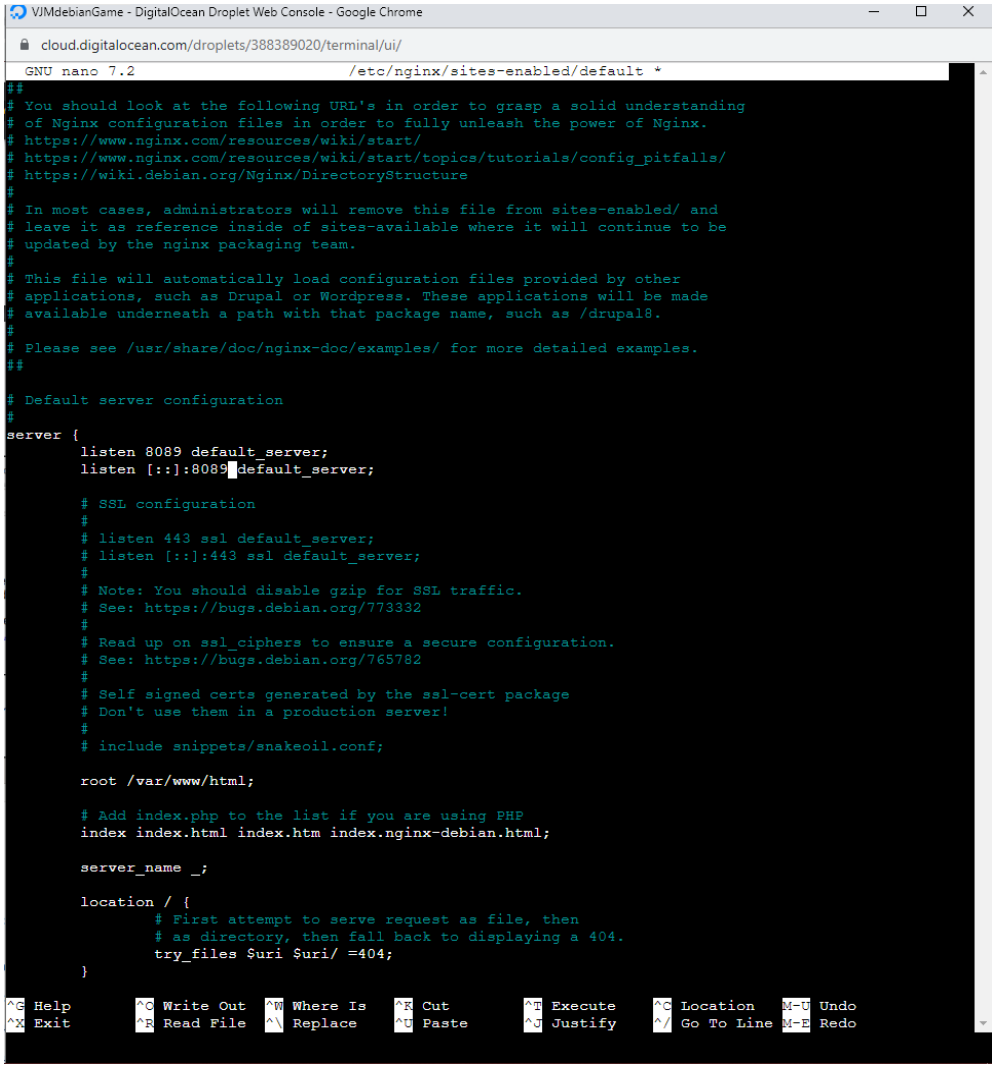


2. Open the SSH configuration file

You can do this by using a text editor to open your “ /etc/ssh/sshd_config” file. We will be using nano.



Change the default port to your desired number, we chose 8089.



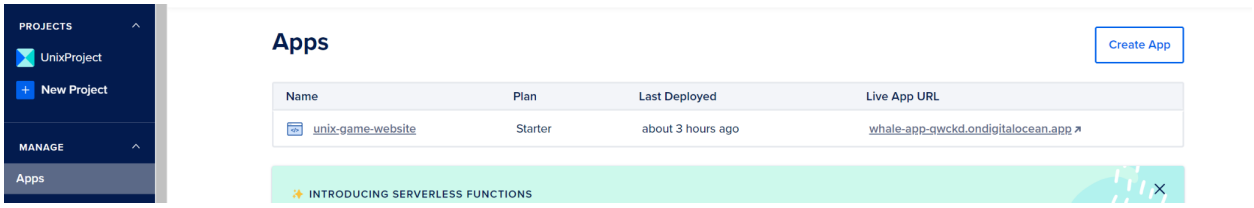
Deploying The Website With an App

We decided to deploy our website through the use of an app, we thought that this was the same thing as hosting the website on our vps. The advantages of the app are that it automatically pulls any changes that we make to our git repository, however, it does not provide a console and we were not aware of this when we decided to use it.

Steps to Deploy the Website With An App:

1. Click on Create App

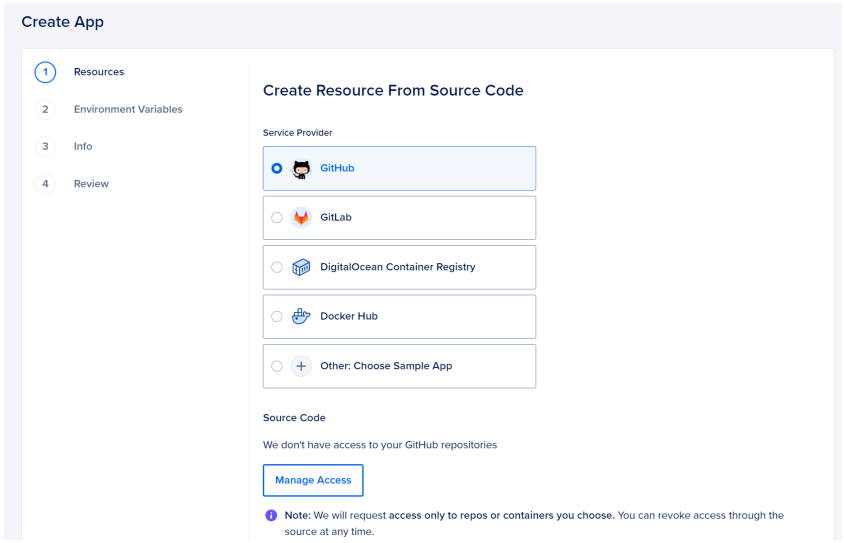
Select “App” under the “Manage” menu on the left side of the screen of your digital ocean account. Click on “Create App”.



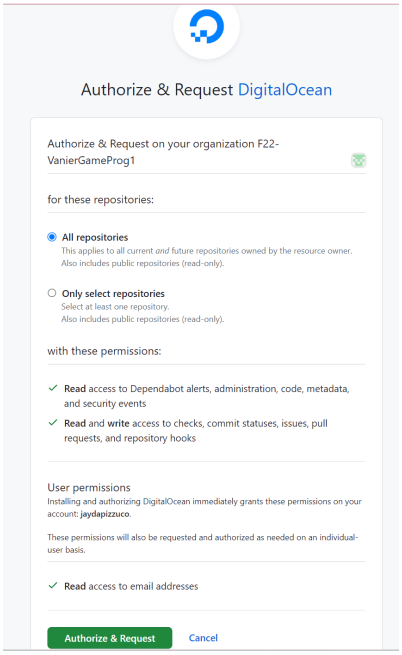
2. Select your GitHub Repository as the Source Code for your App

You will be asked to select a source code for your app, we chose our GitHub repository because this is where all of our work is; the webpages are all in the repository.

Repository: https://github.com/yumvinegar/Site-for-Unix



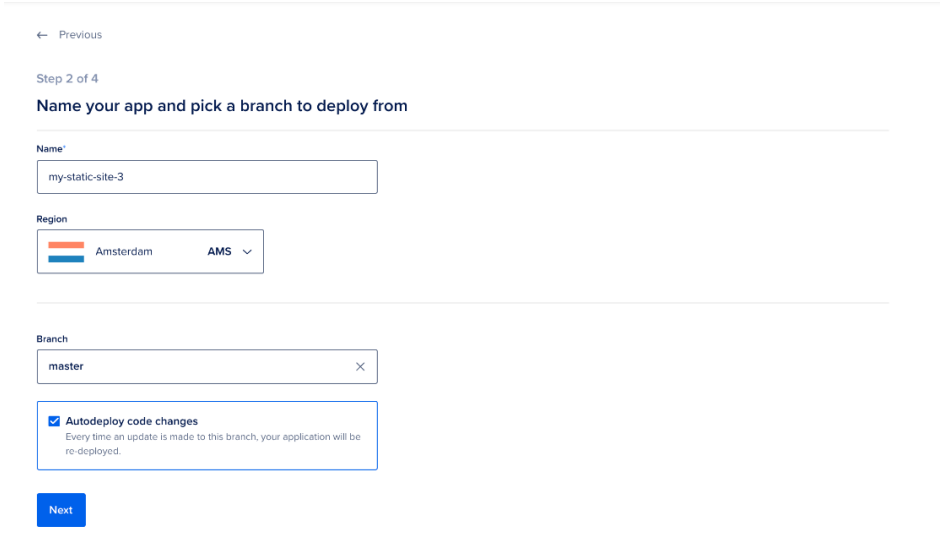
Select GitHub and click on “Manage Access” and you will be asked to choose your repository. You will be asked to authorize Digital Ocean to access your repository, give them the requested authorization.



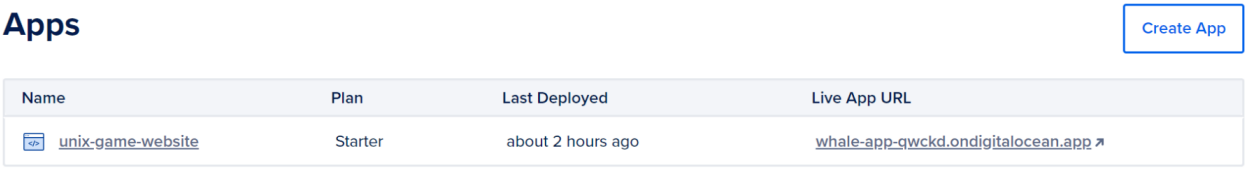
3. Naming Your App and selecting the branch to deploy

You will be asked to name your app, please note that the app name is not the same as the domain name, you must pay a fee to change the url of your website once it has been deployed.

Another option that is provided with apps is the auto deployment of any code changes, this means that any pushes made to the repository will be automatically pulled and will take effect on your website.



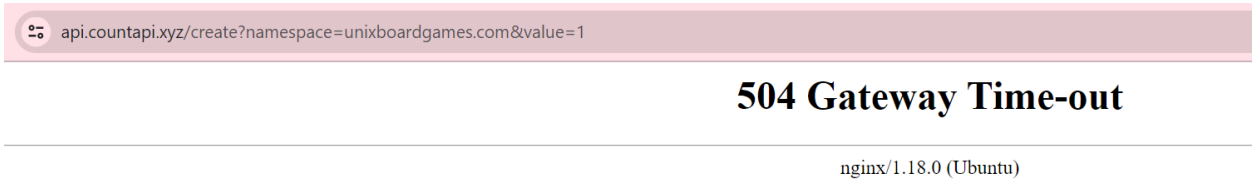
After this step, your app is completed and your website has been deployed. Our app is pictured below.



Automated Task

Our initial aim for our automated task was to create a global visit counter that would be incremented each time someone visited our site. The plan was to keep track of all of the visits and to display this number on the site; all users would see when another user has accessed the site.

Our first strategy to implement our view counter was through the use of a counting service called count api(https://countapi.xyz/).Count Api can be implemented through the use of scripting languages like javascript, JSON and jQuery. It allows you to execute a query and to get the number of matches for that query; it can thus be used to count the amount of total visits to a webpage. However, you must create a namespace and a key to use count api. We tried to do this multiple times and each time we were unable to create our key and we were mean with a 504 error.



Upon accepting the fact that Count Api was not going to work, we began to research and experiment with python scripting. We familiarize ourselves with concepts of python scripting with the help of the 12th Lecture slides entitled “Python Scripting” (Tassia Camoes Araujo).

Ultimately we decided to implement the counter through the use of javascript because we had experimented with it a lot during this semester for our Internet Programming course. Our next strategy to implement the view counter consisted of incrementing a counter variable each time the website page was loaded and then storing this value in the user’s localStorage. This strategy appeared to work at first, however, upon closer examination there were flaws with it. We had initially thought that this method would be able to provide a global count for everyone who visits the website. This is not the case because local storage is local to the device/ browser of the user.

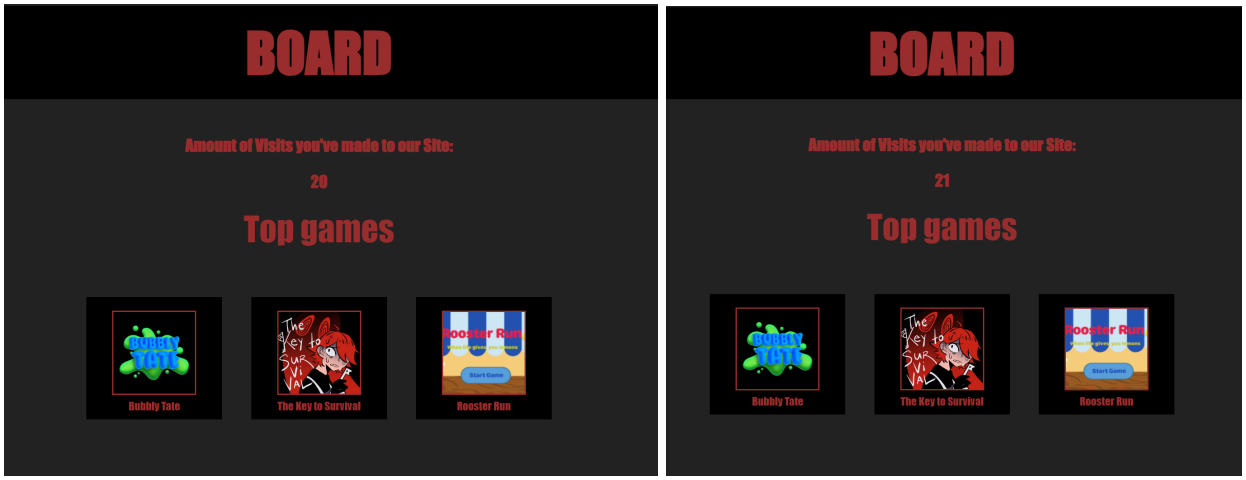
Unfortunately this was not the only issue, because our strategy involved instantiating the counter by using “localStorage.getItem(‘visits’)”, the NaN exception would be displayed on the webpage and the counter did not work. In order for the local Storage method to work, the counter would have to be

initialized with the value 0 upon the first load of the webpage and then it would be incremented by getting from local storage. We tried to implement this but it proved to be very complicated so we decided to look for a better solution.

```
<script>
    var visitsCounter = parseInt(localStorage.getItem("visits"));

    window.onload = function (){

       visitsCounter = visitsCounter + 1;
       localStorage.setItem('visits', visitsCounter);


        visitsCounter = localStorage.getItem('visits');
        document.getElementById('visits').innerHTML = visitsCounter ;
    }
```

Our final strategy and our solution for the view counter was to use cookies. The updatePageVisistsCount() is called when the webpage is loaded. This method checks if the cookie already exists on the webpage and if it does not then it is created and its value is set to one. The count value is then parsed into an int value so that we can increment it in the next step. The cookie is then set with the new count value and this value is displayed on the webpage.

```
<script>

    function updatePageVisitsCount() {
        // Check if the cookie exists
        var countCookie = document.cookie.replace(/(?:(?:^|.*;\s*)loadCount\s*=\s*([^;]*).*$)|^.*$/, "$1");

        // Parse the count from the cookie
        var count = parseInt(countCookie) || 0;

        // Increment the count
        count++;

        // Set the cookie with the new count (expires in 365 days)
        document.cookie = 'loadCount=' + count + '; max-age=' + (365 * 24 * 60 * 60);

        // Display the count on the webpage
        document.getElementById('visits').innerText = count;
    }

     updatePageVisitsCount();
</script>
```



Our view counter does not count the total amount of views for the website as we had originally intended it to. But it does count the amount of visits that each person makes to the website.

Scheduled Process / Service Management

Since we deployed our website on the App function of DigitalOcean, we did not have access to any consoles. Therefore, we realized at almost the end of our process that we should have in fact put the website on the vps (droplet). However, we had already prepared the files and the script to run the scheduled process, which was changing the background color of the website.

Here is the file *ChangeColor.js* with the javascript code to change the color of the background into a random color:

ChangeColor.js

```
function changeColor() {

  var randomColor = '#' + Math.floor(Math.random() * 16777215).toString(16);

  document.body.style.backgroundColor = randomColor;
}

changeColor();
```

Here is the file *change_color.sh* with the script to automatically execute the previous javascript file:

change_color.sh

```
#!/bin/bash

# Navigate to the directory containing your HTML file
cd /var/www/html/index.html

# Execute a JavaScript file to change the background color
node ChangeColor.js
```

Here are the commands that we would have had to run in the console of the vps to effectively execute the script:

Commands to run

```
chmod +x change_color.sh
chmod +x ChangeColor.js

crontab -e

//Example crontab entry to change background color every 5 minutes
*/5 * * * * /path/to/change_color.sh
```

All of these files can be found in the repository in the folder *UnixScript*.