NILE.JS

Torres Baldi Tehuel, Frutos Gastón, Botta Santiago

1er cuatrimestre 2018
Laboratorio de Redes y Sistemas Operativos
UNQ

1. Introducción

1.1. Descripción del proyecto

El proyecto tiene como objetivo principal llevar a cabo una correcta instalación de una aplicación de streaming en algún ordenador y poder utilizarla. Este documento permitirá al usuario realizar la instalación tanto en una computadora local como en una remota.

Se espera que los usuarios puedan realizar streaming de audio y video en vivo, desde diferentes computadoras, utilizando tecnologías peer to peer desde un navegador web.

1.2. Relevamiento de características de las computadoras utilizadas

- CPU: Dual core Intel Core i7-7500U (-HT-MCP-) cache: 4096 KB
  Kernel: 4.4.0-128-generic x86_64 (64 bit gcc: 5.4.0)
  Desktop: Cinnamon 3.2.7 (Gtk 2.24.30)
- CPU: Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz
  Kernel: 4.4.0-128-generic x86_64 (64 bit gcc: 5.4.0)
- MacBook Pro (Retina, 13”, Early 2015)
  CPU: Intel Core i5 - 5287U @ 2.9GHz
  RAM: 8GB DDR3 @ 1867 MHz

1.3. Relevamiento del Sistema operativo

- Sistemas operativos utilizados:
  - Ubuntu 16.04 LTS
  - Linux Mint 18.1 Serena
  - macOS High Sierra 10.13.5

1.4. Software utilizado y/o necesario

- Google Chrome

2. NileJs - Breve descripción

Nile.js[1] es una librería de código abierto (publicado con una licencia MIT de Software Libre) que nos provee un servicio de streaming de video peer-to-peer. Esta librería utiliza

[1] https://github.com/gitsummore/nile.js

servicios proveídos por WebTorrent[2], un protocolo distribuido de entrega de archivos inspirado en BitTorrent y construido con WebRTC.

3. Dependencias

3.1. NVM

Es recomendable utilizar el administrador de versiones de Node (nvm) para correr aplicaciones Node, ya que algunas esperan ser ejecutadas en un ambiente o versión de Node específica. Con nvm podemos administrar fácilmente la versión actual de Node.

Instalación de nvm

Mac:

Para instalar NVM usamos el gestor de paquetes Homebrew[3].

Instalación de Homebrew:

```
1. /usr/bin/ruby -e "$(curl -fsSL
   https://raw.githubusercontent.com/Homebrew/install/master
   /install)"
```

Instalación de nvm:

```
1. brew update
2. brew install nvm
```

Modificar archivo .profile (puede ser ~/.bash_profile, ~/.bashrc, o ~/.zshrc según el shell del usuario) para cargar nvm en el entorno shell. Agregar las siguientes lineas:

```
1. export NVM_DIR=~/.nvm
2. source $(brew --prefix nvm)/nvm.sh
```

Linux:

Instalar un compilador de C++:

```
1. apt-get update
2. apt-get install build-essential libss1-dev
```

Instalar nvm:

```
1. curl-o-https://raw.githubusercontent.com/creationix/nvm/v
   0.33.8/install.sh | bash
```

[2] https://github.com/webtorrent/webtorrent
[3] https://brew.sh/

```
2. source ~/.profile
```

Cambiar versión de Node

Para cambiar de versión de Node con nvm, primero debemos descargar la versión que queremos con el siguiente comando:

```
1. nvm install {número de versión, por ejemplo 4.0}
```

Luego elegir cuál queremos usar:

```
1. nvm use {número de versión, por ejemplo 4.0}
```

3.2. NPM

Npm es el administrador de paquetes de Node, provee las herramientas necesarias para instalar, borrar o modificar las dependencias de un proyecto en Node, así como también ejecutar los scripts de instalación, comienzo, tests, etc, de un proyecto.

Se utilizará Npm o Yarn para instalar las dependencias del proyecto y correr la aplicación.

Instalación de Npm

Mac:

La instalación de NPM se hace automáticamente cuando se instala usando nvm. Si esto no sucede, se puede instalar directamente node (que incluye npm) usando Homebrew. Para esto el comando es

```
1. brew install node
```

Linux:

```
1. sudo apt-get install npm
```

4. Instalación de Nile.JS

Se puede descargar el código desde el repositorio oficial del proyecto Nile.js.

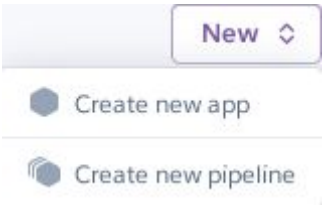
En Node, las dependencias de un proyecto se encuentran declaradas en el archivo package.json, en el root del proyecto. Utilizaremos npm para instalar los paquetes de node necesarios, corriendo el siguiente script en el mismo directorio donde se encuentra el archivo *package.json*:

```
1. npm install
```

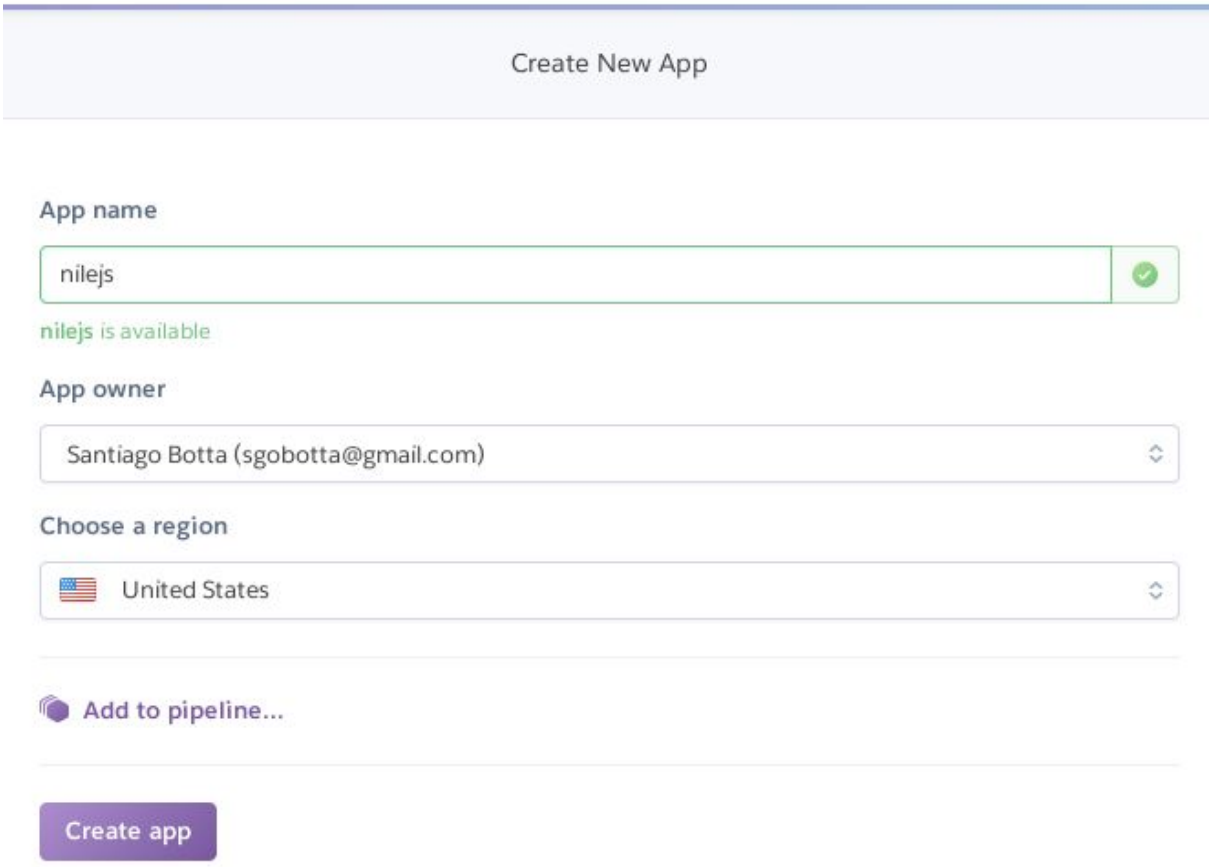
5. Hosting

Luego de haber realizado pruebas localmente decidimos poner a prueba la aplicación remotamente. Previamente habíamos elegido Github para administrar nuestro propio proyecto de NileJs, ésto nos facilitó realizar el hosting en Heroku[4]. Los pasos son simples:

1. Primero fue necesario crear una cuenta en Heroku. Luego elegimos la opción para crear una nueva aplicación.

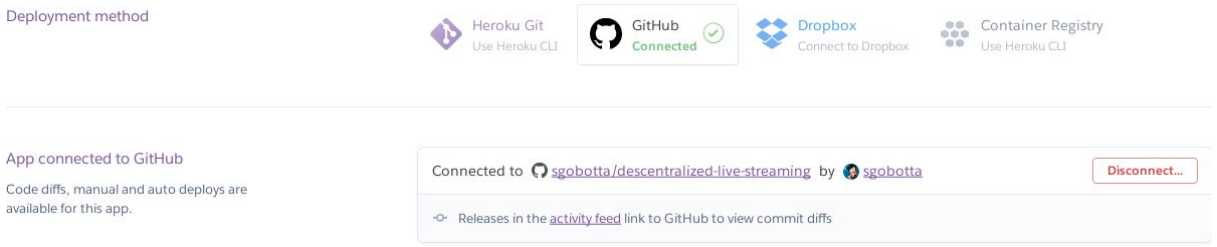


2. A continuación nos aparecerá un cuadro donde debemos elegir un nombre, un owner y la región donde va a estar alojado el servidor.

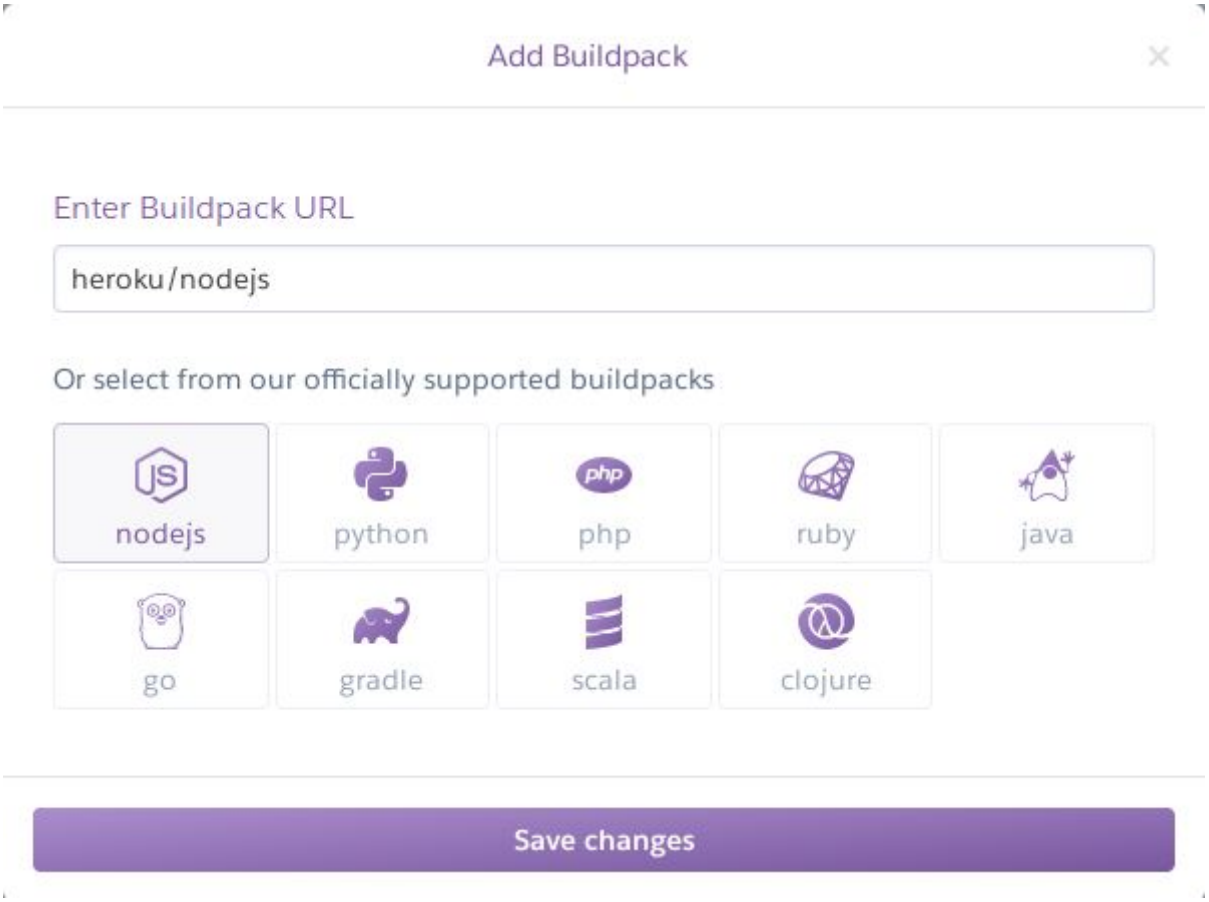


[4] https://www.heroku.com/

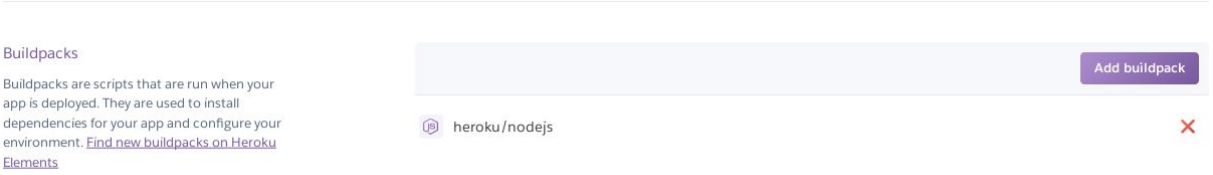
3. El próximo paso es sincronizar el repositorio de Github con nuestra App en Heroku. Para ellos deberíamos encontrarnos en el dashboard de nuestra Heroku App, luego accedemos a la pestaña **Deploy**, donde debemos elegir el repositorio que queremos asociar a nuestra cuenta de Heroku.



4. En algunos casos esto es todo lo que se necesita para poder realizar un deploy de nuestra aplicación en un servidor de Heroku. Sin embargo, nosotros vamos a realizar un paso más para dejar asentado qué tipo de aplicación estamos deployando, en éste caso una aplicación en NodeJs. Para ello existen buildpacks y Heroku nos proporciona una buena cantidad, entre ellos un buildpack para Node apps.
En la pestaña **Settings** se encuentra la configuración con respecto a los métodos de build a utilizar, necesario para realizar el deploy.



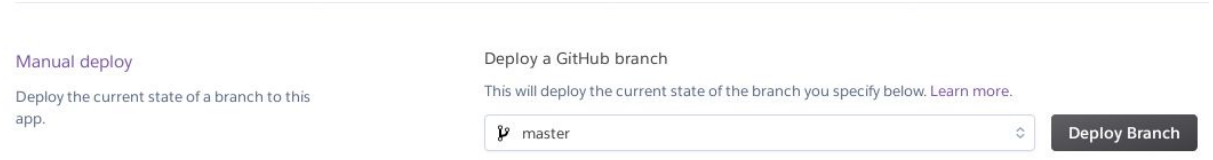
Al guardar los cambios deberíamos visualizar algo de éste estilo:



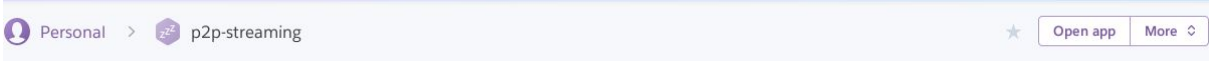
5. Particularmente, el repositorio de NileJs tiene dos archivos de configuración donde se declaran las dependencias, ***package.json*** y ***yarn.lock***. Dado que existe un archivo de configuración de yarn, Heroku utilizará éste para instalar las dependencias en el servidor, durante el deployment.
Es por eso que, luego de instalar las dependencias con npm (*npm install*) debemos ejecutar el comando ***yarn install*** para generar un archivo ***yarn.lock*** válido para Heroku. Éste archivo debe encontrarse versionado en el repositorio Git para que Heroku pueda reconocer las dependencias e instalarlas.

6. Deploy Manual/Deploy Automático

Heroku nos proporciona dos opciones para realizar un deployment, automáticos y manuales desde la pestaña **Deploy**.
Los automáticos se realizan a través de eventos disparados desde Github cada vez que se actualiza una rama. Heroku nos permite especificar desde qué rama se realizarán los deploy automáticos.
Los deploy manuales se pueden realizar en cualquier momento y desde cualquier rama. Nosotros utilizamos ésta última opción, dado que no estamos realizando modificaciones en el repositorio, por el momento.



7. Se abrirá un cuadro de logs, donde es posible visualizar el progreso del proceso de build. Al finalizar podremos abrir una nueva pestaña y visualizar la aplicación, desde el botón que aparece en la parte superior derecha de Heroku.



6. Uso del Framework

El framework viene integrado con un servidor de pruebas donde es posible elegir el rol del usuario, es decir, si es un broadcaster (va a transmitir video) o un viewer.

Para correr el servidor, simplemente ejecutamos el script de inicialización

```
1. npm start
```

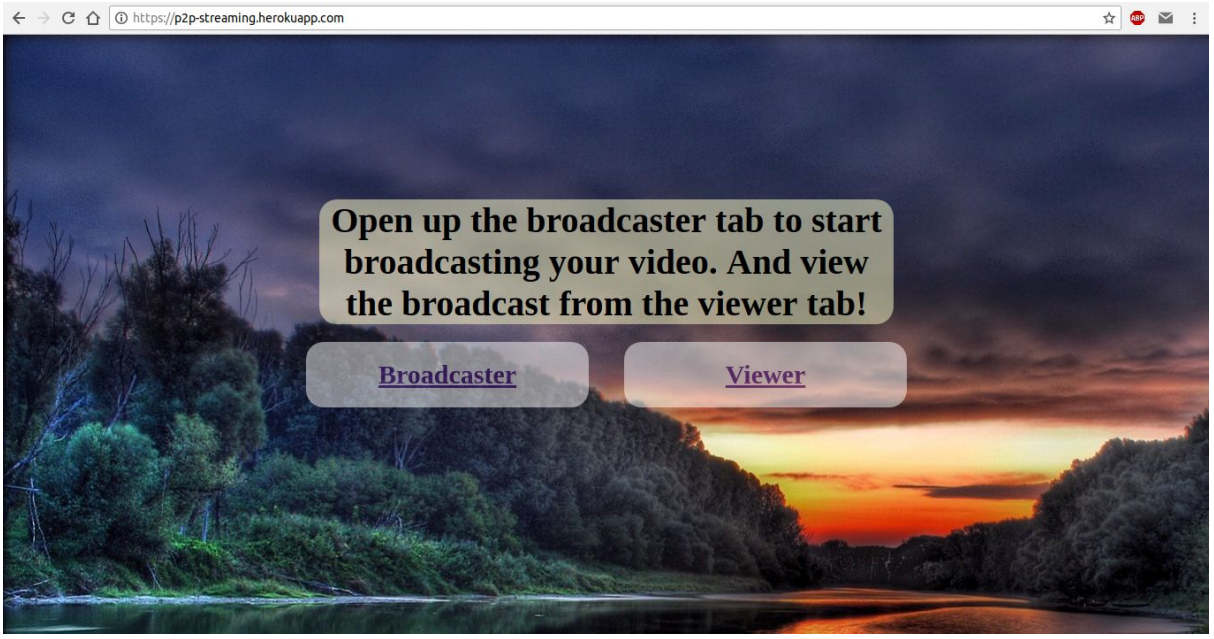
Es posible especificar el puerto donde queremos levantar el server, declarando una variable de ambiente:

```
1. PORT=8001 npm start
```

`Ahora podemos abrir la aplicación en cualquier navegador con http://localhost:8000 o el puerto elegido anteriormente.`

6.1 Home

Al inicio podemos ver una pantalla que nos da para elegir si queremos stremear o ser espectadores.

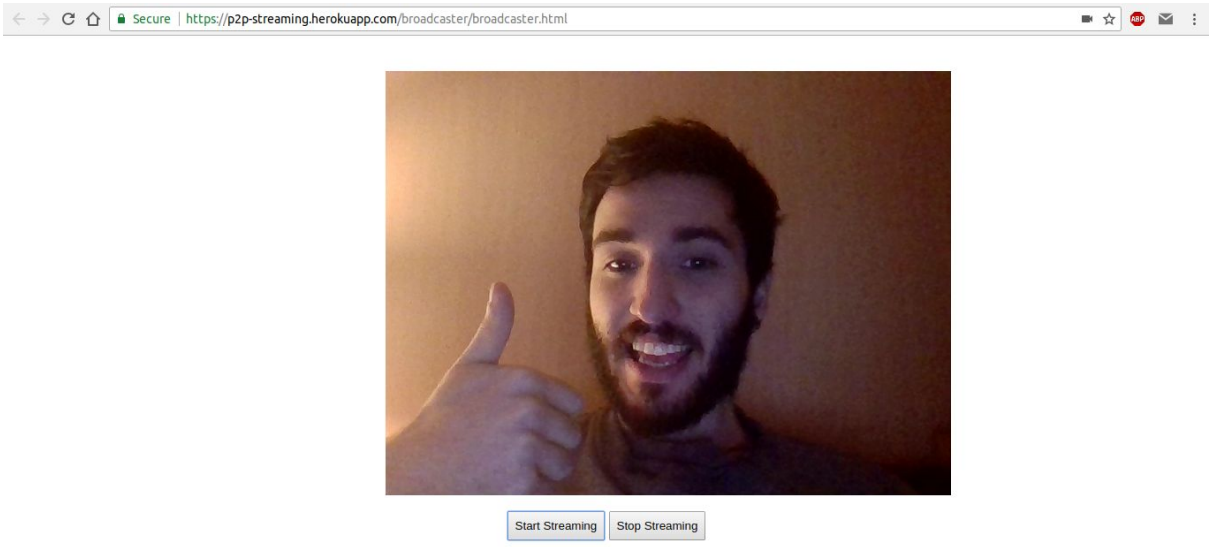


6.2 Broadcaster

Al seleccionar Broadcaster, podemos ver dos botones:

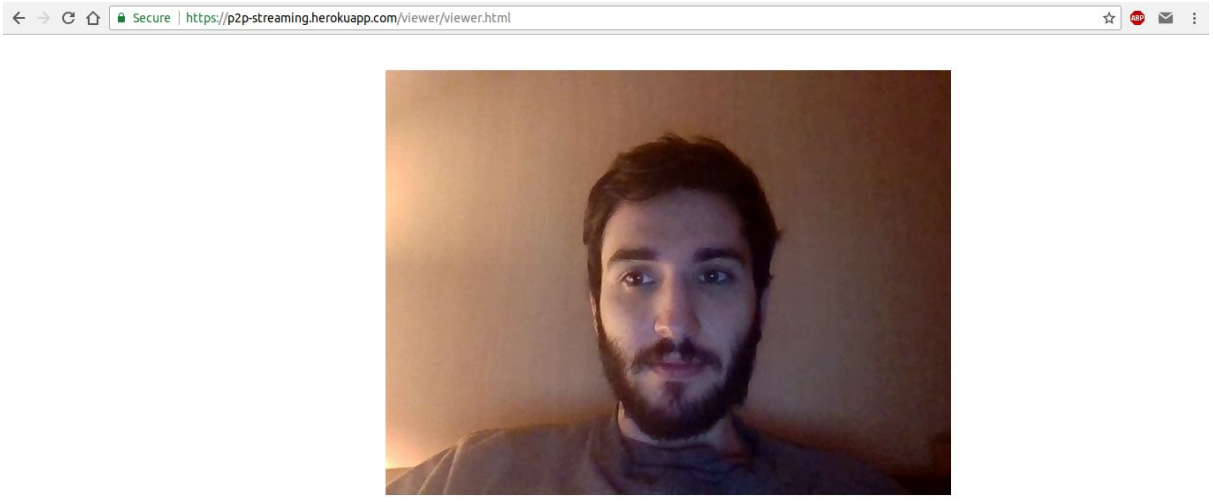
1. El botón de comenzar sl streaming
2. El botón de parar el streaming.

Al presionar el botón de comenzar, nos pedirá permisos para utilizar la webcam y, al aceptarlo, veremos lo que se está grabando y el streaming ya estará online.



6.3 Viewer

Al seleccionar la opción de viewer, iremos a una sección donde, si hay un streaming en línea, podremos verlo en la pantalla.

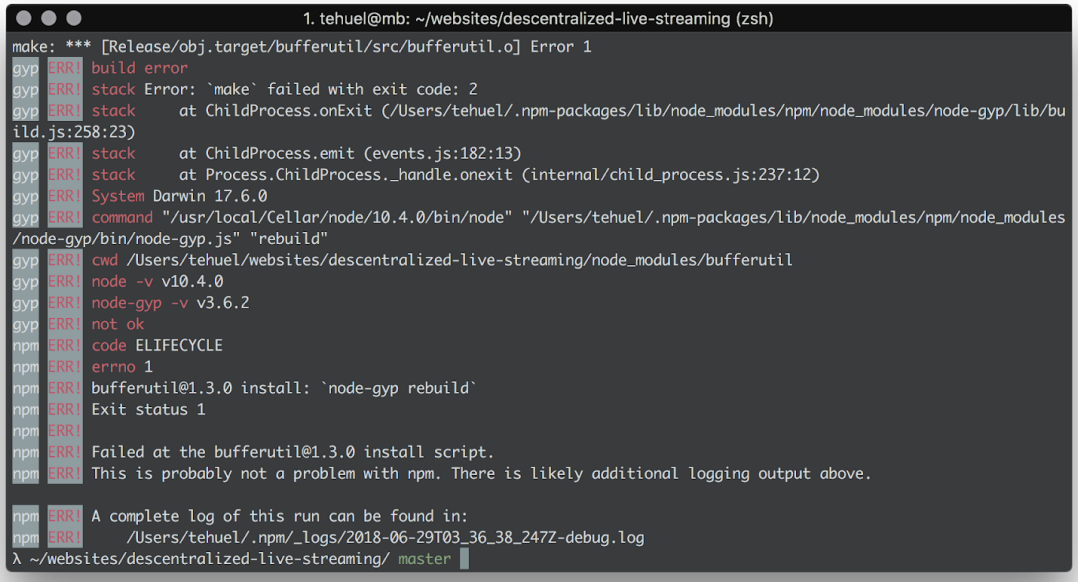


7. Problemas surgidos

Problemas en macOS para instalar dependencias del proyecto usando NPM

Tuvimos un error al momento de ejecutar el comando npm install, para descargar las dependencias del proyecto.

El error estaba relacionado con la ejecución del comando “node-gyp rebuild”. Este comando era ejecutado internamente por npm.



Investigando un poco en internet nos encontramos con varios issues de github con información relevante al error:

1. Otro proyecto diferente (node.bcrypt) en el que se presentaba el mismo problema[5]
    a. En este issue en particular la solución más votada[6] consistia en borrar algunas carpetas generadas por el programa node-gyp, y volver a ejecutar npm install. Esto no funcionó.
2. Otro error más general de node[7], donde el problema se le atribuye a cambios internos que hubo entre las versiones 8 y 10 de node.

[5] https://github.com/kelektiv/node.bcrypt.js/issues/297
[6] https://github.com/kelektiv/node.bcrypt.js/issues/297#issuecomment-264457169
[7] https://github.com/nodejs/node/issues/20770

3. Un tercer issue interesante[8], donde se discuten detalles de la implementación de algunos de esos cambios entre las versiones 8 y 10 de node, que pueden ocasionar problemas con paquetes no preparados para estos cambios.

La solución finalmente fue la utilización de NVM, para instalar la version 8 de node, en la que el comando npm install se pudo ejecutar correctamente.

Adblock Plus

Teniendo el complemento adblock plus[9] instalado en el navegador al momento de hacer las pruebas, la aplicación mostraba por la salida de la consola errores al no poder conectarse a algunos trackers que eran utilizados para descargar información de los torrents. Esto impedía el correcto funcionamiento de la aplicación.

La solución fue deshabilitar el complemento adblock plus al momento de hacer las pruebas.

Problemas con las redes de la UNQ

Al momento de hacer pruebas usando la conexión de internet de la universidad, no logramos conseguir que la aplicación funcione correctamente.

Asumimos que esto es debido a algún bloqueo o límite existente en la configuración de las redes de la universidad, probablemente relacionado con el protocolo torrent.

No pudimos identificar el problema en particular, ya que la salida por consola de la aplicación no indicaba ningun error, pero al mismo tiempo no pudimos ver ningún video siendo transmitido.

La solución a esto fue usar la conexión a internet compartida desde alguno de nuestros teléfonos celulares.

8. Referencias

[1] - https://github.com/gitsummore/nile.js

[2] - https://github.com/webtorrent/webtorrent

[3] - https://brew.sh/

[4] - https://www.heroku.com/

[5] - https://github.com/kelektiv/node.bcrypt.js/issues/297

[6] - https://github.com/kelektiv/node.bcrypt.js/issues/297#issuecomment-264457169

[8] https://github.com/nodejs/nan/issues/504
[9] https://adblockplus.org/es/

[7] - https://github.com/nodejs/node/issues/20770

[8] - https://github.com/nodejs/nan/issues/504

[9] - https://adblockplus.org/es/