

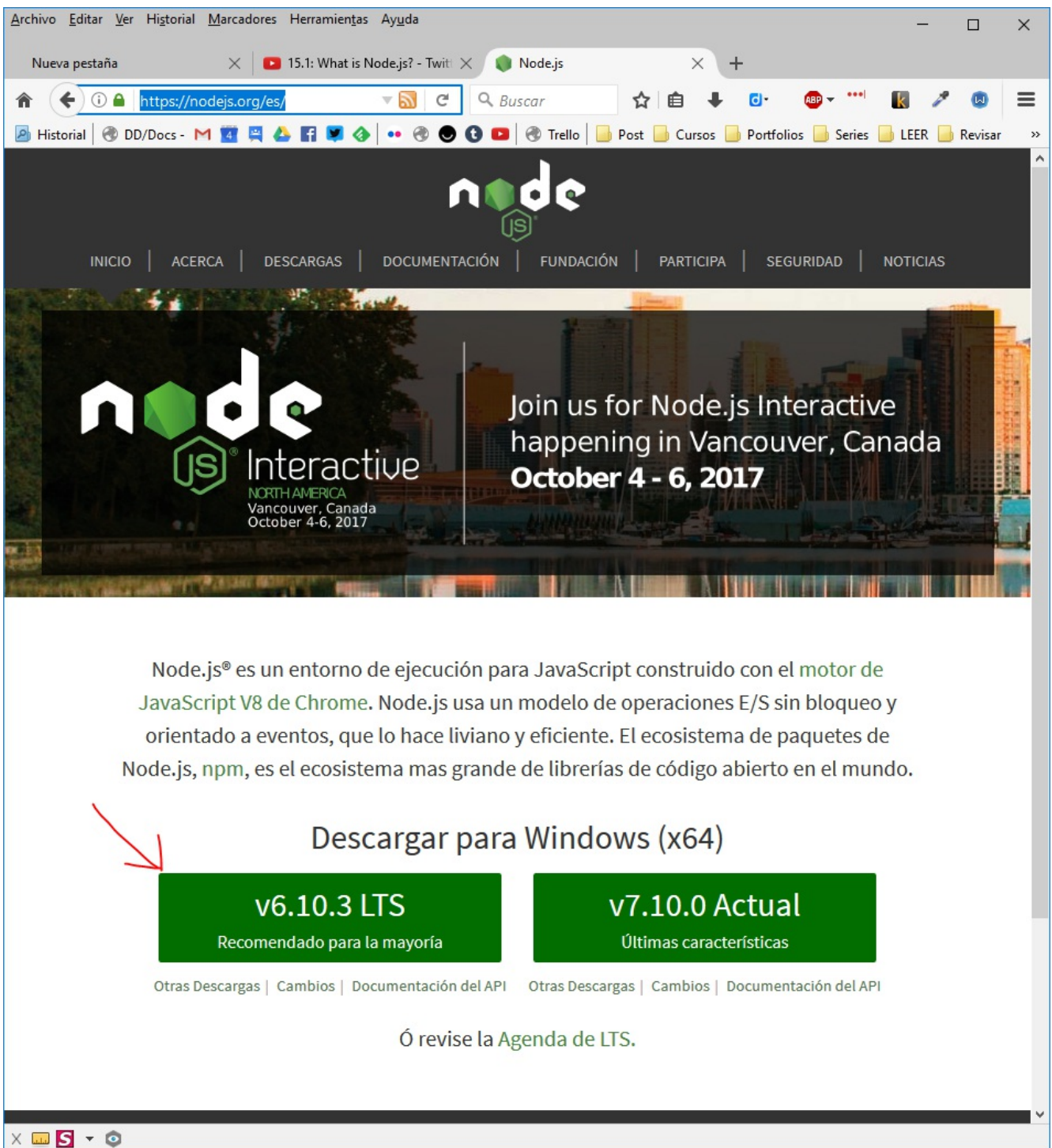
TwitterBot paso a paso

Referencias

- Tutorial D Shiffman <https://youtu.be/s70-Vsud9Vk?list=PLRqwX-V7Uu6atTSxoRiVnSuOn6JHnq2yV>
- Node.js <https://nodejs.org/es/>
- NPM <https://www.npmjs.com/>
- Twit package <https://www.npmjs.com/package/twit>
- Twitter platform <https://dev.twitter.com/>
- Twitter APIs <https://dev.twitter.com/overview/api>
- Amazon WS <https://aws.amazon.com/es/>
- Bot <https://twitter.com/n074b07>

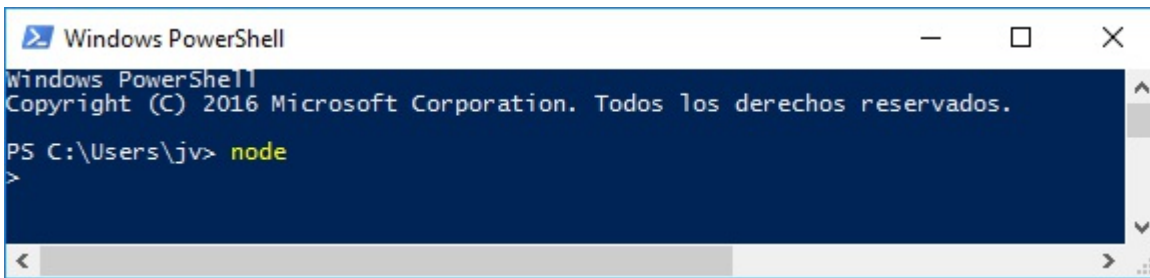
1.- Instalar Node.js

<https://nodejs.org/es/>



2.- Ejecutar Node.js

En PowerShell o Terminal



```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos los derechos reservados.

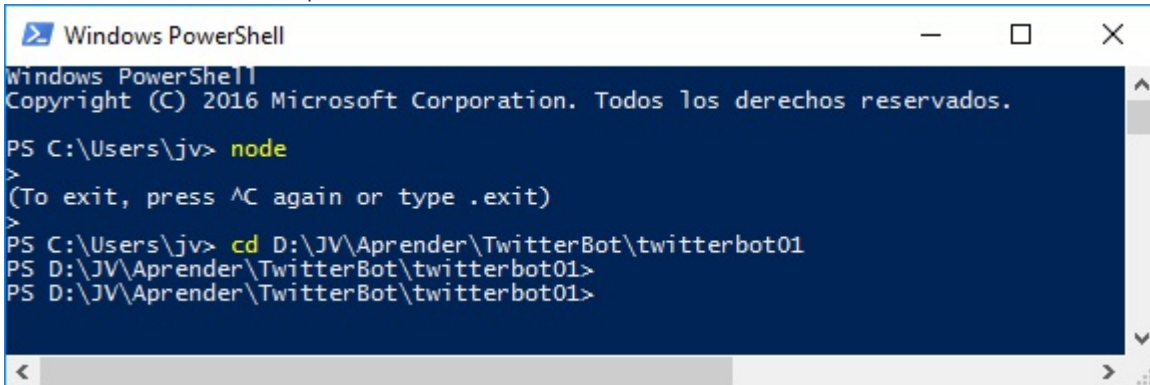
PS C:\Users\jv> node
>
```

3.- Creamos carpeta y archivo bot.js

- twitterbot01/ -- bot.js

4.- Instalamos el paquete Twit usando Node Package Manager

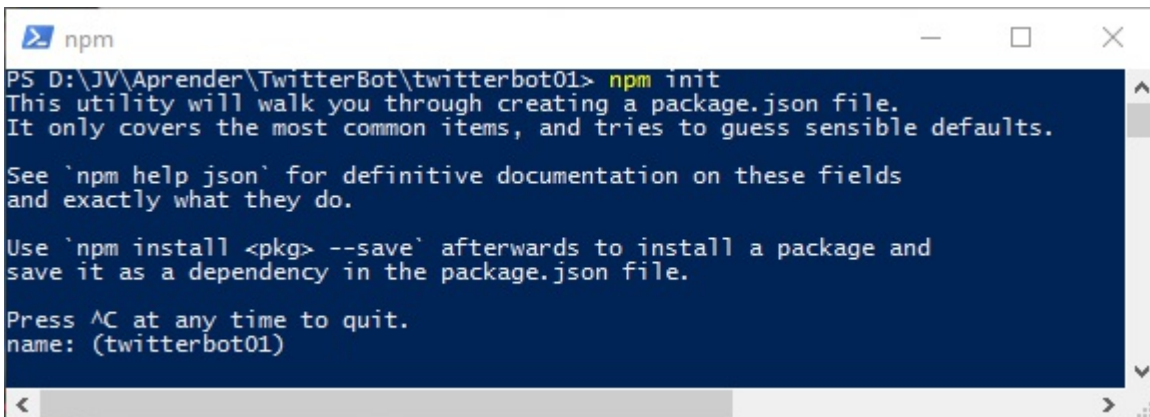
Abrimos en el terminal la carpeta del bot



```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\jv> node
>
(To exit, press ^C again or type .exit)
>
PS C:\Users\jv> cd D:\JV\Aprender\TwitterBot\twitterbot01
PS D:\JV\Aprender\TwitterBot\twitterbot01>
PS D:\JV\Aprender\TwitterBot\twitterbot01>
```

npm init



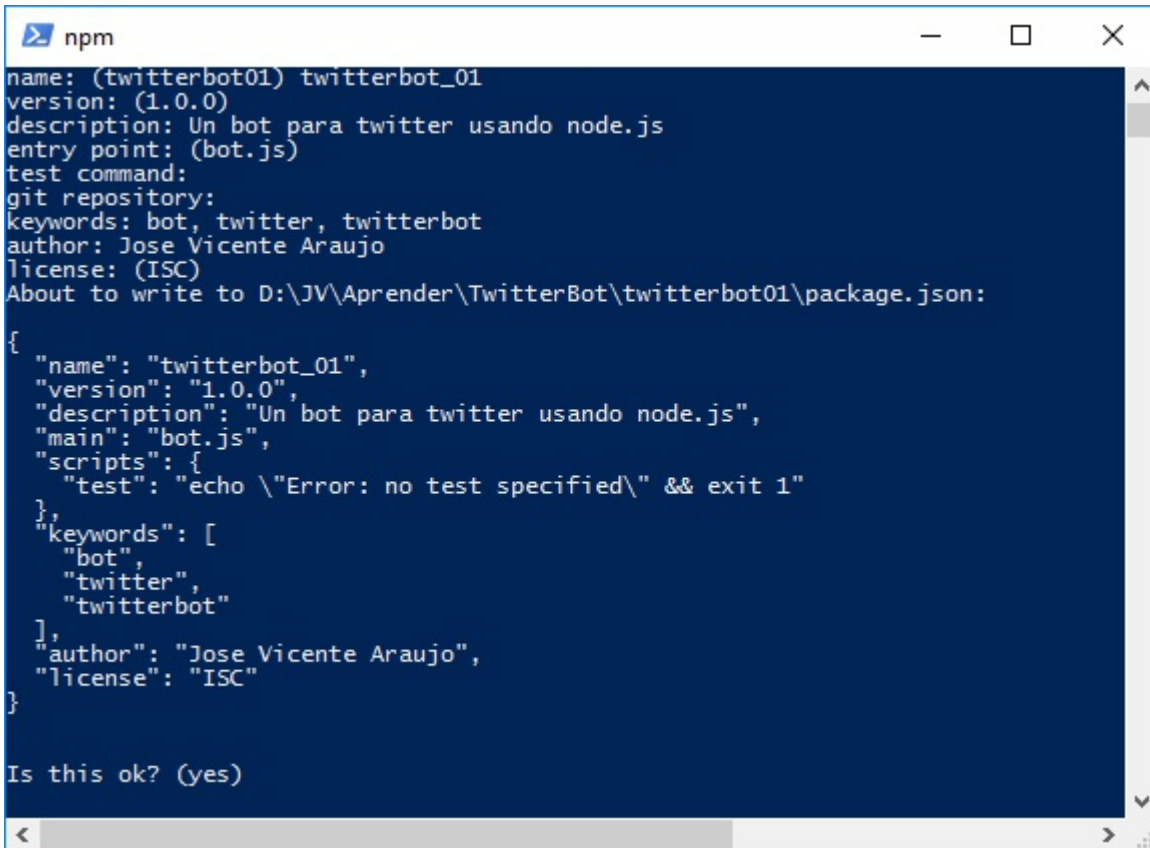
```
npm
PS D:\JV\Aprender\TwitterBot\twitterbot01> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg> --save' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (twitterbot01)
```

Datos del paquete



```
npm
name: (twitterbot01) twitterbot_01
version: (1.0.0)
description: Un bot para twitter usando node.js
entry point: (bot.js)
test command:
git repository:
keywords: bot, twitter, twitterbot
author: Jose Vicente Araujo
license: (ISC)
About to write to D:\JV\Aprender\TwitterBot\twitterbot01\package.json:

{
  "name": "twitterbot_01",
  "version": "1.0.0",
  "description": "Un bot para twitter usando node.js",
  "main": "bot.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "bot",
    "twitter",
    "twitterbot"
  ],
  "author": "Jose Vicente Araujo",
  "license": "ISC"
}

Is this ok? (yes)
```

NPM creará el archivo *package.json* en la carpeta especificada.

Instalamos Twit: `npm install twit --save`

```

Windows PowerShell
PS D:\JV\Aprender\TwitterBot\twitterbot01> npm install twit
twitterbot_01@1.0.0 D:\JV\Aprender\TwitterBot\twitterbot01
-- twit@2.2.5
+-- bluebird@3.5.0
+-- mime@1.3.4
+-- request@2.81.0
+-- aws-sign2@0.6.0
+-- aws4@1.6.0
+-- caseless@0.12.0
+-- combined-stream@1.0.5
|   -- delayed-stream@1.0.0
+-- extend@3.0.1
+-- forever-agent@0.6.1
+-- form-data@2.1.4
|   -- async@2.6.1
+-- har-validator@4.2.1
|   +-- ajv@4.11.8
|   |   +-- co@4.6.0
|   |   |   -- json-stable-stringify@1.0.1
|   |   |   -- jsonify@0.0.0
|   |   -- har-schema@1.0.5
+-- hawk@3.1.3
|   +-- boom@2.10.1
|   +-- cryptiles@2.0.5
|   +-- hoek@2.16.3
|   |   -- sntp@1.0.9
+-- http-signature@1.1.1
|   +-- assert-plus@0.2.0
|   +-- jsprim@1.4.0
|   |   +-- assert-plus@1.0.0
|   |   +-- extsprintf@1.0.2
|   |   +-- json-schema@0.2.3
|   |   |   -- verror@1.3.6
|   |   -- sshpk@1.13.0
|   |   +-- asn1@0.2.3
|   |   +-- assert-plus@1.0.0
|   |   +-- bcrypt-pbkdf@1.0.1
|   |   +-- dashdash@1.14.1
|   |   |   -- assert-plus@1.0.0
|   |   +-- ecc-jsbn@0.1.1
|   |   +-- getpass@0.1.7
|   |   |   -- assert-plus@1.0.0
|   |   +-- jodid25519@1.0.2
|   |   +-- jsbn@0.1.1
|   |   |   -- tweetnacl@0.14.5
+-- is-typedarray@1.0.0
+-- isstream@0.1.2
+-- json-stringify-safe@5.0.1
+-- mime-types@2.1.15
|   -- mime-db@1.27.0
+-- oauth-sign@0.8.2
+-- performance-now@0.2.0
+-- qs@6.4.0
+-- safe-buffer@5.0.1
+-- stringstream@0.0.5
+-- tough-cookie@2.3.2
|   -- punycode@1.4.1
+-- tunnel-agent@0.6.0
-- uuid@3.0.1

npm WARN twitterbot_01@1.0.0 No repository field.
PS D:\JV\Aprender\TwitterBot\twitterbot01>

```

Esto crea la carpeta `node_modules` y modifica `package.json` para incluir el paquete `Twit` en `dependencies`.

5.- Editamos `bot.js`

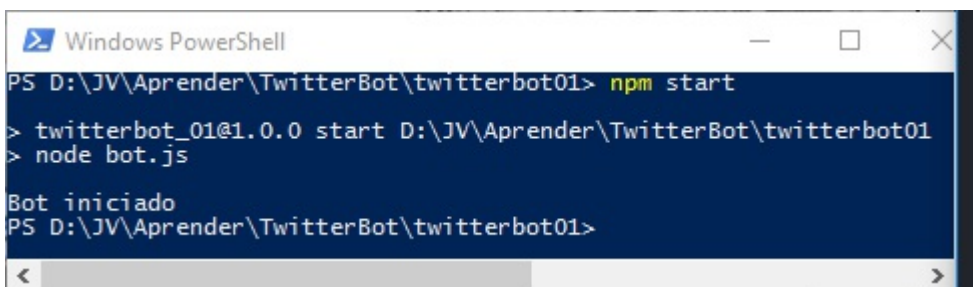
De momento simplemente añadiremos esto:

```
console.log("Bot iniciado");
```

6.- Añadimos *bot.js* a *package.json*

```
{
  "name": "twitterbot_01",
  "version": "1.0.0",
  "description": "Un bot para twitter usando node.js",
  "main": "bot.js",
  "scripts": {
    "start": "node bot.js",
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [
    "bot",
    "twitter",
    "twitterbot"
  ],
  "author": "Jose Vicente Araujo",
  "license": "ISC",
  "dependencies": {
    "twit": "^2.2.5"
  }
}
```

Comprobamos en la consola que la orden `npm start` funciona



```
Windows PowerShell
PS D:\JV\Aprender\TwitterBot\twitterbot01> npm start

> twitterbot_01@1.0.0 start D:\JV\Aprender\TwitterBot\twitterbot01
> node bot.js

Bot iniciado
PS D:\JV\Aprender\TwitterBot\twitterbot01>
```

7.- Volvemos a editar *bot.js* para importar Twit y autenticarnos en Twitter

Localizamos los detalles y ejemplos del paquete Twit en su repositorio: <https://github.com/ttezel/twit>

Añadimos: `var Twit = require('twit');`

Para autenticarnos en Twitter:

```
var T = new Twit({
  consumer_key: '...',
```

```

consumer_secret: '...',
access_token: '...',
access_token_secret: '...',
timeout_ms: 60*1000, // optional HTTP request timeout to apply to all requests.
})

```

Cambiamos los puntos suspensivos por los datos que obtendremos en <https://dev.twitter.com>

Solutions	Products	Resources	Tools	Community
Build	Publisher platform	Documentation	API status	#TapIntoTwitter
Customer service	REST APIs	Forums	API console	Official partner
Build great apps	Streaming APIs	Blog	Cards validator	Events
Tell great stories	Ads API	Case studies	Manage my apps	Flight 2015
	Gnip	Developer terms		Flight 2014
		Policy support		

Clic en **Create New App**

Rellenar todos los campos y aceptar. En la pestaña **Keys and Access Tokens**, copiar las claves requeridas sustituyendo los puntos suspensivos.

Clic en **Create my access token** y copiar también las nuevas claves

8.- Publicar un mensaje

Añadimos esto a bot.js:

```

T.post('statuses/update', { status: 'hello world!' }, function(err, data, response) {
  console.log(data)
})

```

9.- Obtener los resultados de una búsqueda

Añadimos:

```

T.get('search/tweets', { q: 'banana since:2011-07-11', count: 100 }, function(err, data, response) {
  console.log(data)
})

```

Otra opción menos compacta:

```

var params = {
  q: 'glitch', //qué buscamos
  count: 5 //cuántos resultados
}

T.get('search/tweets', params, gotData);

function gotData(err, data, response){
  var tweets = data.statuses;
  for (var i = 0; i < tweets.length; i++){
    console.log(tweets[i].text);
  }
}

```

```
}

```

10.- Programar publicaciones

```
tweetIt(); //para enviar antes de empezar a contar el intervalo
setInterval(tweetIt, 1000*60*60*24) //24 horas

function tweetIt(){
  var r = Math.floor(Math.random()*100);

  var tweet = {
    status: 'Test msg #' + r
  }

  T.post('statuses/update', tweet, tweeted);

  function tweeted(err, data, response) {
    if(err) {
      console.log("¡Error al tuitear!");
    } else {
      console.log(data)
    }
  }
}
```

11.- Responder a eventos

Por ejemplo cuando alguien me sigue:

```
//Setting up a user stream
var stream = T.stream('user');

//Cada vez que alguien me sigue
stream.on('follow', followed);

function followed(eventMsg) {
  var name = eventMsg.source.name;
  var screenName = eventMsg.source.screen_name;
  tweetIt('@' + screenName + ' gracias por seguirme');
}

function tweetIt(txt){
  var r = Math.floor(Math.random()*100);

  var tweet = {
    status: txt
  }

  T.post('statuses/update', tweet, tweeted);

  function tweeted(err, data, response) {
    if(err) {
      console.log("¡Error al tuitear!");
    } else {
      console.log(data)
    }
  }
}
```

Como se ve, hemos añadido un parámetro a la función *tweetIt()* para poder personalizar los mensajes que envía

12.- Ejecutar un sketch de Processing desde Node.js y tuitear el resultado

Vamos a dividirlo en partes

Creamos la estructura del sketch

En la misma carpeta donde tenemos el bot creamos otra llamada **sketch**, y dentro de ella un archivo **sketch.pde**. El programa debe producir una imagen llamada **output.png**

Creamos una función que ejecute el sketch y tuitee la imagen

```
//Ejecutar la función
tweetP5Img();

function tweetP5Img(txt){
  //Ejecutar el sketch de Processing
  //Descomentar para ejecutar en local
  var cmd = 'processing-java --sketch=D:/JV/Aprender/TwitterBot/twitterbot01/sketch --run';
  //Descomentar para ejecutar en el servidor
  //var cmd = 'sketch/sketch';

  //Ejecutar el subprocesso
  exec(cmd, processing);

  //Subproceso
  function processing(){
    var filename = 'sketch/output.png';
    var params = {
      encoding: 'base64'
    }
    var b64 = fs.readFileSync(filename, params);

    //Subir imagen
    T.post('media/upload', {media_data: b64}, uploaded);

    //Una vez subida, tuitear imagen
    function uploaded(err, data, response){
      var id = data.media_id_string;
      var tweet = {
        status: '#glitchIt',
        media_ids: [id]
      }

      //Tuitear mensaje + imagen
      T.post('statuses/update', tweet, tweeted);
    }
  }

  //Callback
  function tweeted(err, data, response){
    if(err){
      console.log("Algo ha fallado");
    } else {
      console.log("¡Yuju!");
    }
  }
}
```

Preparamos el servidor para ejecutar Processing

En este supuesto usamos **Amazon EC2**, pero debería funcionar de manera parecida en cualquier servidor Linux

- Vamos a <https://aws.amazon.com/es/ec2/>
- **Consola de administración de AWS:** https://console.aws.amazon.com/?nc2=h_m_mc
- Identificarse
- Ir a **EC2**
- **Launch Instance**
- Seleccionar **Ubuntu Server 16.04**
- Seleccionar **t2.micro** (Free tier eligible)
- **Review and Launch**
- **Launch**

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair name
twitterbot01

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel

Launch Instances

Abriendo twitterbot01.pem

Ha elegido abrir:

twitterbot01.pem

que es: Text Document

de: https://us-west-2.console.aws.amazon.com

¿Qué debería hacer Firefox con este archivo?

☐ Abrir con

Bloc de notas (predeterminada)

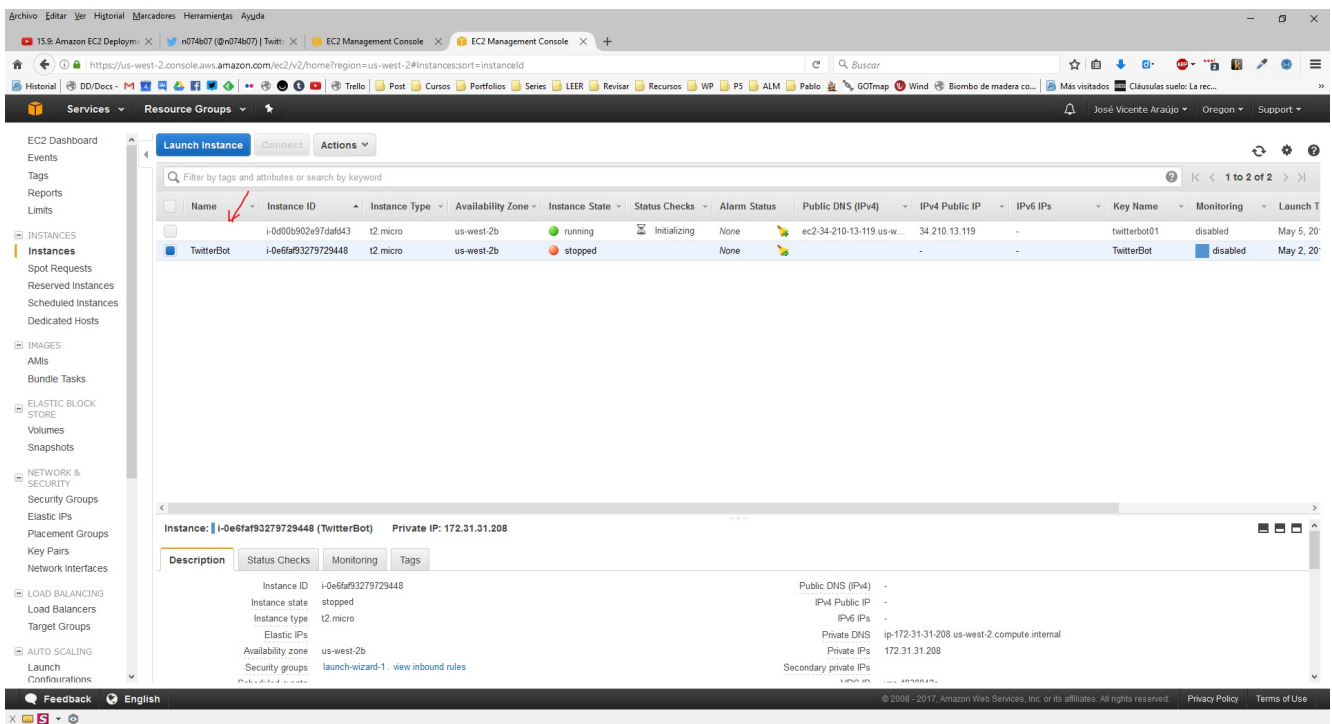
☒ Guardar archivo

☐ Hacer esto automáticamente para estos archivos a partir de ahora.

Aceptar

Cancelar

- Launch instances
- View Instances

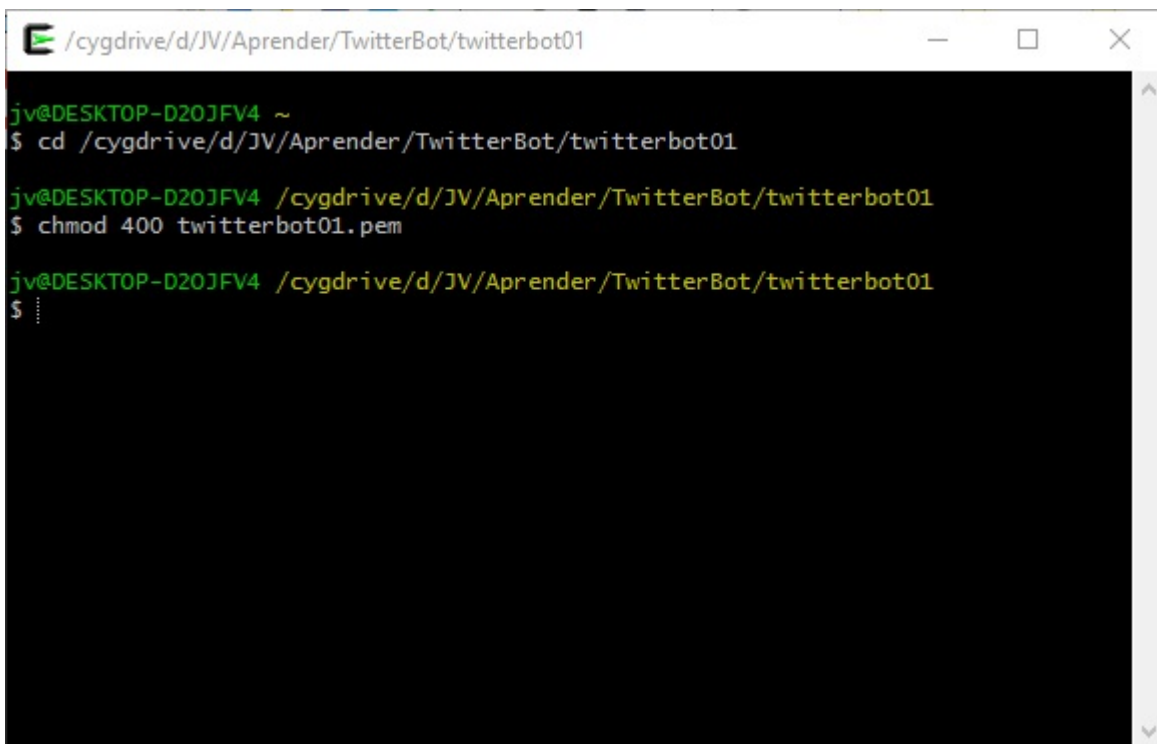


- Nombrar la instancia **twitterbot01** (por ejemplo)
- Clic derecho / clic Connect / Copiar el texto de ejemplo para acceder vía SSH

Preparamos el archivo .pem

Vamos a necesitar cambiar permisos y acceder al servidor vía SSH, para lo que será conveniente usar **Cygwin** y asegurarnos de instalar el componente **openssh**.

- Movemos el archivo .pem a la carpeta del bot.
- Abrimos Cygwin y nos dirigimos a la carpeta del bot
- `chmod 400 twitterbot01.pem`



```
/cygdrive/d/JV/Aprender/TwitterBot/twitterbot01

jv@DESKTOP-D20JFV4 ~
$ cd /cygdrive/d/JV/Aprender/TwitterBot/twitterbot01

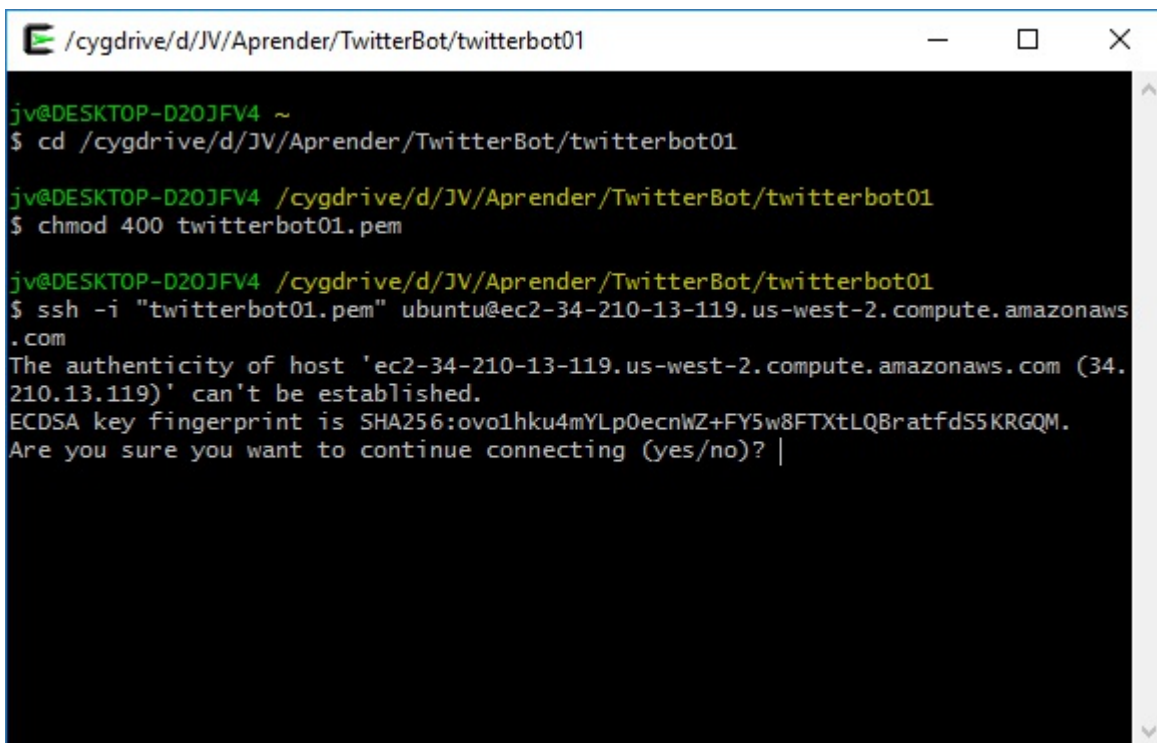
jv@DESKTOP-D20JFV4 /cygdrive/d/JV/Aprender/TwitterBot/twitterbot01
$ chmod 400 twitterbot01.pem

jv@DESKTOP-D20JFV4 /cygdrive/d/JV/Aprender/TwitterBot/twitterbot01
$
```

Accedemos al servidor vía SSH

En Cygwin. Si da error, instalar **openssh**

- `ssh -i "twitterbot01.pem" ubuntu@ec2-34-210-13-119.us-west-2.compute.amazonaws.com` (lo copiamos en el último paso del apartado anterior)



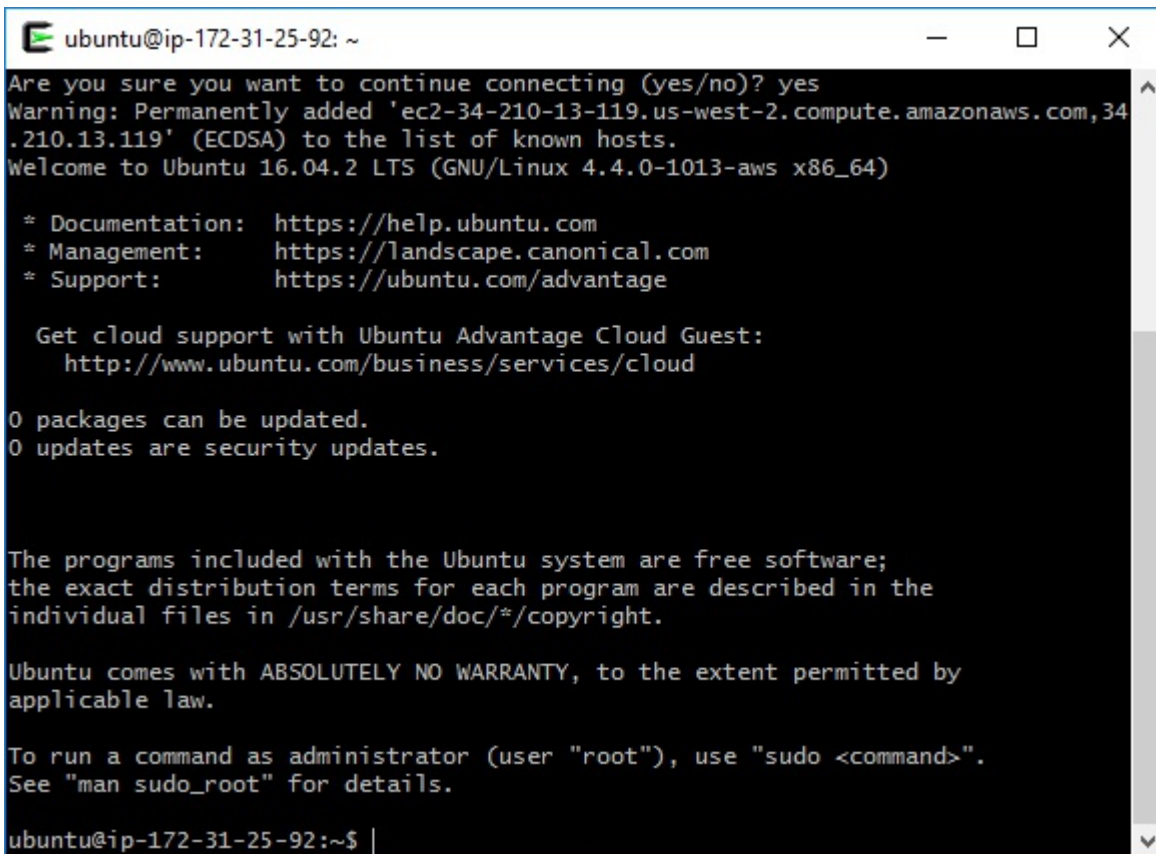
```
/cygdrive/d/JV/Aprender/TwitterBot/twitterbot01

jv@DESKTOP-D20JFV4 ~
$ cd /cygdrive/d/JV/Aprender/TwitterBot/twitterbot01

jv@DESKTOP-D20JFV4 /cygdrive/d/JV/Aprender/TwitterBot/twitterbot01
$ chmod 400 twitterbot01.pem

jv@DESKTOP-D20JFV4 /cygdrive/d/JV/Aprender/TwitterBot/twitterbot01
$ ssh -i "twitterbot01.pem" ubuntu@ec2-34-210-13-119.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-34-210-13-119.us-west-2.compute.amazonaws.com (34.210.13.119)' can't be established.
ECDSA key fingerprint is SHA256:ovo1hku4mYLp0ecnwZ+FY5w8FTXtLQBratfd55KRGQM.
Are you sure you want to continue connecting (yes/no)? |
```

- `yes`

A terminal window titled 'ubuntu@ip-172-31-25-92: ~' with standard window controls. The terminal output shows a connection confirmation, a warning about host keys, a welcome message for Ubuntu 16.04.2 LTS, links for documentation, management, and support, information about Ubuntu Advantage Cloud Guest, package update status, and system information. The prompt 'ubuntu@ip-172-31-25-92:~\$' is visible at the bottom.

```
ubuntu@ip-172-31-25-92: ~
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-34-210-13-119.us-west-2.compute.amazonaws.com,34.210.13.119' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-1013-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

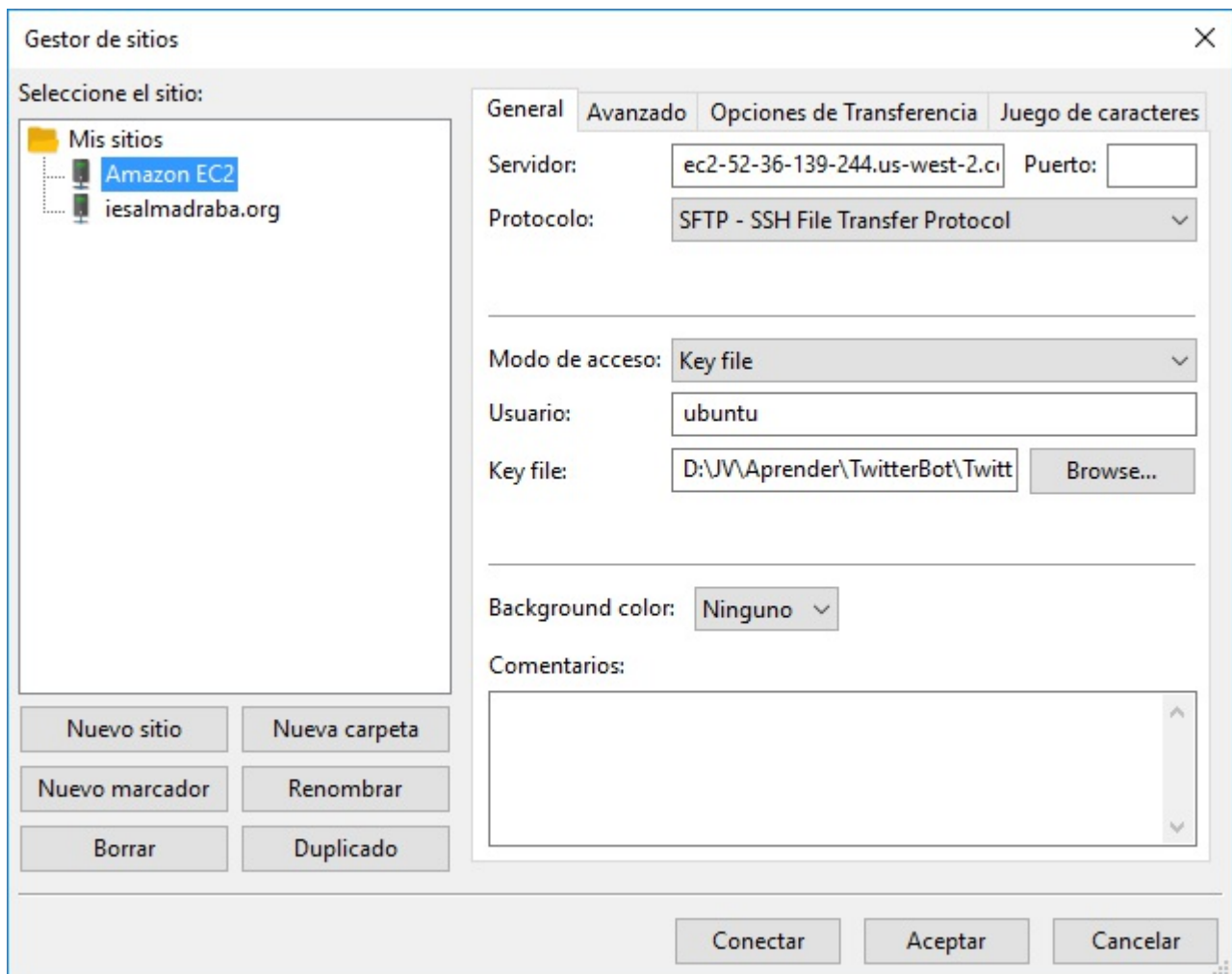
ubuntu@ip-172-31-25-92:~$ |
```

Instalar Node.js en el servidor vía SSH

- `sudo apt-get update`
- `sudo apt-get install nodejs`
- `sudo apt-get install npm`
- `sudo ln -s `which nodejs` /usr/bin/node`

Subimos los archivos del bot vía FTP

- Abrimos **FileZilla**



- Subir los archivos **bot.js** y **package.json** a la carpeta **ubuntu**

Instalamos los módulos requeridos (Twit)

- `npm install`

Instalamos Java

- `sudo apt-get install default-jre`

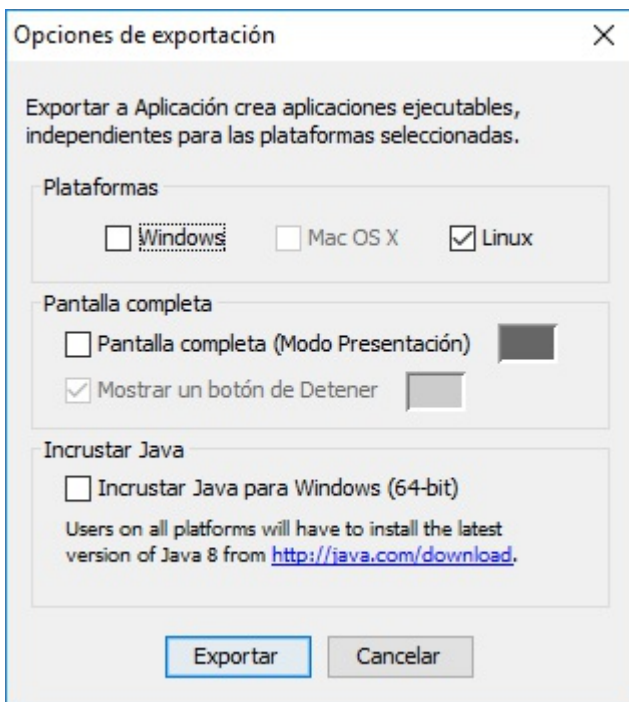
Instalamos un monitor virtual

Processing necesita ejecutarse en una ventana, pero nuestro servidor no tiene interfaz gráfico. Instalaremos uno virtual.

- `sudo apt-get install xvfb libxrender1 libxtst6 libxi6`
- Creamos el display falso: `sudo Xvfb :1 -screen 0 1024x768x24`
- **Ctrl + c**
- Nos aseguramos de que se abre en segundo plano `sudo nohup Xvfb :1 -screen 0 1024x768x24 &`
- `export DISPLAY=":1"`

Exportamos nuestro sketch para linux

- **Archivo / Exportar aplicación**



- renombramos la carpeta **application-linux64** a **sketch**
- La subimos al servidor vía FTP
- Entramos en ella: `cd sketch`
- Cambiamos los permisos del ejecutable: `chmod 777 sketch`
- Lo ejecutamos: `./sketch`
- Comprobamos que se ha ejecutado listando el contenido de la carpeta en busca del archivo **output.png**: `ls`

```

ubuntu@ip-172-31-25-92: ~/sketch
[2]+  Exit 1                  sudo nohup Xvfb :1 -screen 0 1024x768x24
ubuntu@ip-172-31-25-92:~$ export DISPLAY=":1"
-bash: export: `=:1': not a valid identifier
ubuntu@ip-172-31-25-92:~$ sudo nohup Xvfb :1 -screen 0 1024x768x24 &
[2] 24672
ubuntu@ip-172-31-25-92:~$ nohup: ignoring input and appending output to 'nohup.out'
export DISPLAY=":1"
-bash: export: `=:1': not a valid identifier
[2]+  Exit 1                  sudo nohup Xvfb :1 -screen 0 1024x768x24
ubuntu@ip-172-31-25-92:~$ export DISPLAY=:1
ubuntu@ip-172-31-25-92:~$ chmod 777 sketch
ubuntu@ip-172-31-25-92:~$ ./sketch
-bash: ./sketch: Is a directory
ubuntu@ip-172-31-25-92:~$ sketch
The program 'sketch' is currently not installed. You can install it by typing:
sudo apt install sketch
ubuntu@ip-172-31-25-92:~$ ./sketch
-bash: ./sketch: Is a directory
ubuntu@ip-172-31-25-92:~$ cd sketch
ubuntu@ip-172-31-25-92:~/sketch$ ./sketch
-bash: ./sketch: Permission denied
ubuntu@ip-172-31-25-92:~/sketch$ chmod 777 sketch
ubuntu@ip-172-31-25-92:~/sketch$ ./sketch
ubuntu@ip-172-31-25-92:~/sketch$ ls
lib  output.png  sketch  source
ubuntu@ip-172-31-25-92:~/sketch$ |

```

Instalar Forever

Forever mantiene el bot activo incluso si salimos del servidor.

En Cygwin:

- `sudo npm install forever -g`
- `forever start bot.js`
- Para comprobar qué procesos están en marcha: `forever list`
- Si queremos parar un proceso: `forever stop bot.js`
- Si queremos parar todos los procesos: `forever stopall`

Reinicios o actualizaciones

Si fuera necesario reiniciar o actualizar el bot, la secuencia sería:

- Subir los archivos modificados (vía FTP)
- Acceder al servidor vía SSH:


```
ssh -i "twitterbot01.pem" ubuntu@ec2-34-210-13-119.us-west-2.compute.amazonaws.com
```
- `sudo nohup Xvfb :1 -screen 0 1024x768x24 &`
- `export DISPLAY=:1`
- `forever start bot.js`
- `exit`