Trabajo Práctico Final
Deployd

Materia: Laboratorio de Redes y Sistemas Operativos

Profesor: José Luis di Biase

Integrantes: Nicolás Allip
Martin Carniello
Luis Coronel

Índice

1- ¿Qué es Deployd?

Deployd es una herramienta que facilita la construcción de una API, relacionado con la importante funcionalidad fuera del programa que conoce la demanda de aplicaciones complejas.

Convención sobre configuración

En una herramienta como Deployd, su propósito es proveer funcionalidad, se requiere configuración para asegurar el resultado final y se comporte de la manera que usted necesite. No queríamos enviar un archivo de configuración a los usuario. Eso no es inspirador y es tedioso. Cuando estas haciendo una app, deberías sentir que estás creando algo y no solo seleccionando valores.

Algunos frameworks evitan esto permitiendo configurar la app con código. En Deployd, es posible que hayas creado tu API escribiendo una clase que herede de la clase Collection. Esto es mejor que escribir archivos de configuración, pero últimamente conduce a una gran cantidad de código repetitivo. En una convención sobre el entorno de configuración, cuando quieres usar la funcionalidad por defecto, escribes esencialmente un archivo vacío. Cuando quieres anular la funcionalidad por defecto, escribes todo desde cero.

Nuestra solución es el DashBoard. En pocas palabras, es una IDE basada en la web para configurar archivos Deployd. Es una interfaz expresiva que te permite crear tus app en la API visualmente.

Lógica debería ser código

Aunque no queríamos que nuestros usuarios escriban el código para la preparación estructural que está mejor manejado en una interfaz personalizada, nos damos cuenta que no podemos tener usuarios que adhieran lógica de negocio en este camino. La lógica de negocios es diferente para cada app, y no podemos responder por cada caso de uso.

Al mismo tiempo, No podemos forzar un negocio lógico fuera del lado del servidor y sobre el cliente; eso dejaría tu app abierta a ataques de personas que descifrarían tu API.

Decidimos que no hay mejor manera de definir la lógica que por la escritura de código , así que hicimos Deployd scripts. Al escribir eventos o ganchos en la funcionalidad por defecto (como la creación de objetos , su actualización y consulta de ellos) , puede crear casi cualquier aplicación con las reglas : la validación , la seguridad , las optimizaciones de consulta , las relaciones, y más.

Un motor API

Llamamos Deployd algún motor API - porque pensamos que su pariente más cercano es en realidad un motor de videojuegos.

Un motor de juego asume que cada juego tiene niveles (o mapas, o habitaciones , o escenas) que contienen los objetos del juego , que se necesiten hacer a la pantalla e interactuar con los demás. La mayoría de los motores tienen un editor de niveles, donde se puede construir estos niveles sin ningún tipo de generación de código; cuando haya terminado con su nivel , que no es exactamente un juego jugable, pero funciona y se puede caminar alrededor de un mapa que construiste - esta es la creación con configuración más básica vigente.

Del mismo modo, Deployd asume que todas las API a construir cuentan con colecciones de objetos de datos que necesitan para apoyar las operaciones CRUD (crear, leer, actualizar, eliminar) . Las colecciones son creadas en la versión de Deployd de un editor de niveles llamado dashboard.

Siguiendo con la analogía, un motor de juego también sabe que su juego va a necesitar cosas como la detección de colisiones y el movimiento, pero no sabe exactamente cómo desea implementar que, por lo que le permite escribir scripts en puntos estratégicos como el "on collision" u "on tick ".

Dibujar un cohete en la pantalla es realmente similar en casi todo juego, no así lo que ocurre cuando ese cohete golpea a un enemigo, por lo que se puede escribir ese comportamiento en un pequeño script, y el motor del juego lo ejecuta en el momento adecuado. Esta estrategia evita la repetición de código sin quitarle su poder y flexibilidad.

Las partes redundantes de más bajo nivel de Deployd, son el acceso de enrutamiento y base de datos, por lo que los construye en el núcleo y no habría que preocuparse de ellos. La lógica de negocio, la cual es diferente para cada aplicación, se escribe a partir de Eventos - se usan métodos “hooks” sobre las operaciones CRUD.

Para algunas personas, este no es el enfoque adecuado para una aplicación final; prefieren tener un control total sobre el backend para ajustar el rendimiento y la funcionalidad de bajo nivel como mejor les parezca y se sientan incómodos estando limitados a la lógica de negocio. (Muchos desarrolladores de juegos eligen no utilizar un motor de juego y prefieren una biblioteca de renderizado de nivel inferior para prácticamente las mismas razones) En esos casos, se recomienda Deployd como una herramienta de creación de prototipos en lugar de un marco backend completo.

El acercamiento a esta tecnología, es recomendable para los desarrolladores que prefieren pasar el tiempo construyendo interfaces de usuario.

2- Instalando Deployd paso a paso

Referencia: http://docs.deployd.com/docs/getting-started/installing-deployd.html

Versión de SO: Ubuntu 10.04

2.1- Herramientas a utilizar

NodeJS

Node.js es un entorno de programación en la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos.
Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor.

Sitio Oficial: http://nodejs.org/

NPM

npm es el administrador de paquetes para JavaScript. Corre a través de la línea de comando y administra las dependencias de una aplicación. Además, le permite al usuario instalar aplicaciones Node.js que estén disponibles en el registro npm.
npm está escrito enteramente en JavaScript

Sitio Oficial: https://www.npmjs.com/

MongoDB

MongoDB es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Sitio Oficial: http://www.mongodb.org/

2.2- Instalación de NodeJS

1.) Instalación de NodeJS con los paquetes de Ubuntu:

```
sudo apt-get install nodejs
```

2.) chequear la versión de node.js

```
node -v
```

```
v0.10.33
```

2.3- Instalación de NPM

1.) Instalación de NPM de los paquetes de Ubuntu:

```
sudo apt-get install npm
```

2.4- Instalar MongoDB

1.) Para importar la clave pública GPG:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
7F0CEB10
```

2.) Creamos un archivo de lista de mongodb /etc/apt/sources.list.d/mongodb.list

```
echo 'deb
http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist
10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
```

3.) Actualizamos el repositorio

```
sudo apt-get update
```

4.) Instalamos los paquetes de la versión estable de MongoDB

```
sudo apt-get install mongodb-10gen

echo "mongodb-10gen hold" | sudo dpkg --set-selections
```

5.) Chequeamos que haya sido instalado y que versión tiene

```
mongod --version
```

2.5- Instalación Deployd

1.) Para instalar Deployd desde NPM:

```
npm install deployd -g
```

4- Comandos Deployd en terminal

dpd

Inicia el proyecto actual de Deployd en modo desarrollo con una terminal interactiva para interactuar con el servidor en funcionamiento

dpd create [name]

Crea un nuevo proyecto Deployd con el nombre pasado como parámetro

dpd showkey

Imprime la clave de la aplicación para poder usarla en un dashboard remoto o hacer pedidos administrativos con autentificación de manera remota

dpd keygen

Genera una nueva clave para el acceso y administración remoto

dpd remote

Abre el dashboard remoto en el navegador

5- Primera API

Para crear un nuevo proyecto desde el comienzo, se deberá tipear los siguientes comandos en la terminal:

1.) Crear nuevo proyecto, pasándole un nombre como parámetro

```
dpd create hello-world
```

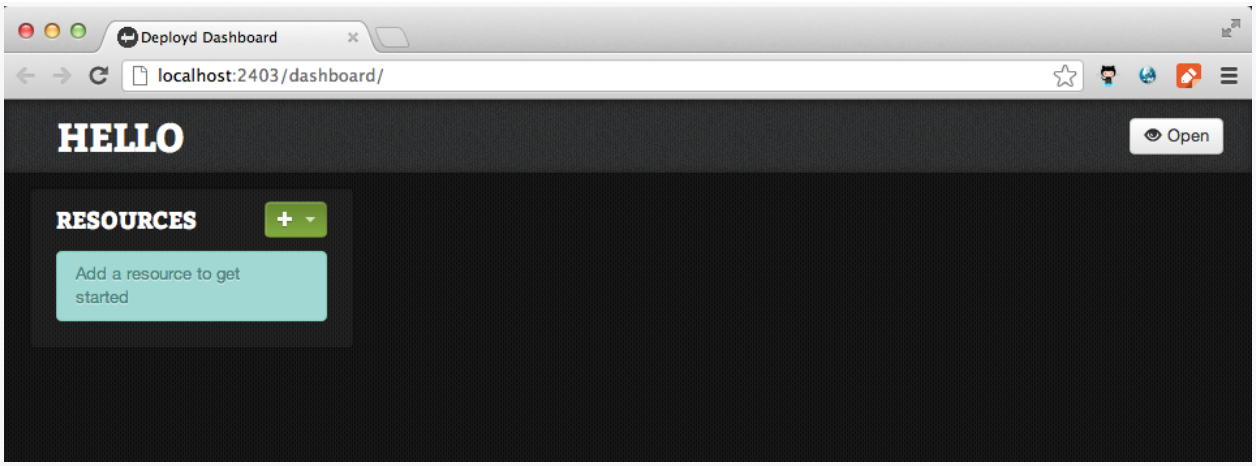
2.) Dirigirse a la carpeta donde se creó el proyecto

```
cd hello-world
```

3.) Ejecutar el servidor Deployd

```
dpd -d
```

Con estos comandos se creará una API, correrá el servidor Deployd y se abrirá el dashboard de la aplicación

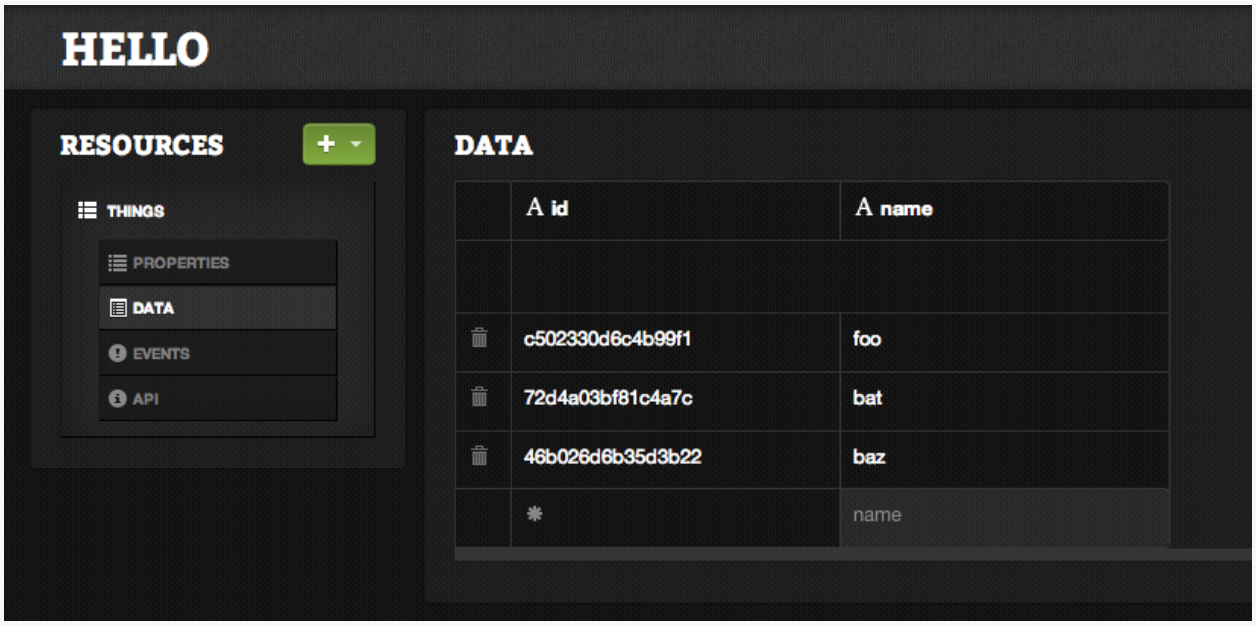


Crear una nueva colección, seleccionandola directamente con el botón “+” que se encuentra a la derecha de “Resources”. En “Create New Collection” escribir el nombre de la colección que se desee (a continuación crearemos la colección “/things”)

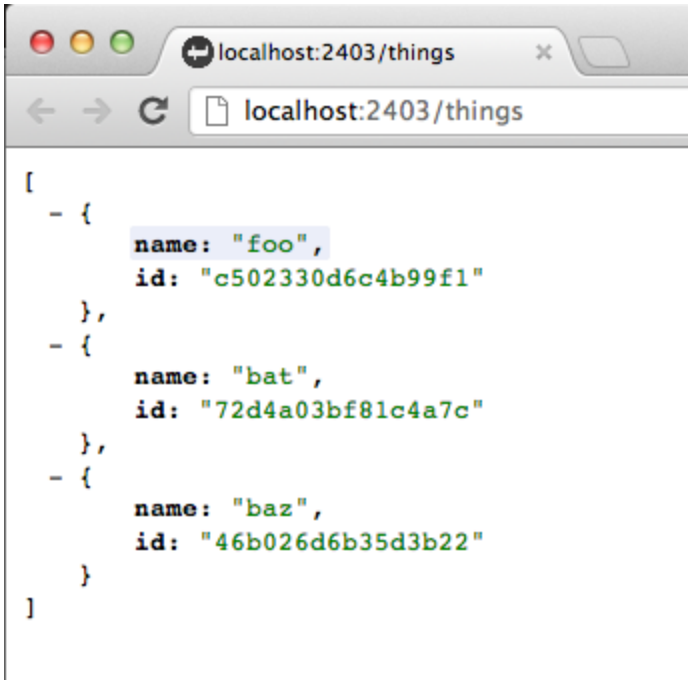
Generaremos una propiedad “name” a la colección recientemente creada. Para esto hay que hacer click en “Properties” y tipear el nombre de la propiedad que queramos, a continuación presionar enter.

Agregaremos algo de información a la colección abriendo el editor de información. Hacer click en “Data”. Tipear algunos nombres, como por ejemplo: “foo” y apretar enter, luego “bat” y apretar enter nuevamente, y por último “baz” y apretar enter.

Se debería ver algo así:



Clickear sobre “API” para ver la documentación para acceder a la colección. Si se accede por medio del navegador a la dirección http://localhost:2403/things, se deberá ver el siguiente JSON:



Abrir en un navegador la dirección http://localhost:2403 y en la consola tipear lo siguiente:

```
dpd.things.get( console.log );
dpd.things.get( { $limit: 1 }, console.log );
dpd.things.get( { name: ‘foo’ }, console.log );
```

Se debería ver lo siguiente:

6- Primera aplicación consumiendo API

Crearemos una aplicación simple desde el principio en Deployd. Se deberá contar con algún conocimiento básico de JQuery para hacerlo.

1.) Para empezar, crearemos una nueva aplicación Deployd desde la línea de comandos:

    dpd create comments

2.) Nos dirigimos a la carpeta principal del proyecto

    cd comments

3.) Reemplazamos el archivo “index.html” de la carpeta “public”:

```
<!DOCTYPE html>
<html>
<head>
    <title>Deployd Tutorial</title>
    <style type="text/css">
        body { font-size: 16pt; }
        .container { width: 960px; margin-left: auto; margin-right: auto; }
        form { border: #cccccc 1px solid; padding: 20px; margin-bottom: 10px;
-moz-border-radius: 5px; -webkit-border-radius: 5px; border-radius: 5px; }
        .form-element { margin-bottom: 10px; }
        #refresh-btn { margin-bottom: 20px; }
        .comment { padding: 10px; margin-bottom: 10px; border-bottom: #cccccc 1px solid; }
        .comment .links { float: right; }
        .comment .links a { margin-left: 10px; }
        .comment .author { font-style: italic; }
    </style>
</head>
<body>
    <div class="container">
        <div id="comments">
```

```
        </div>
        <form id="comment-form">
            <div class="form-element">
                <label for="name">Name: </label>
                <input type="text" id="name" name="name" />
            </div>
            <div class="form-element">
                <textarea id="comment" name="comment" rows="5" cols="50">
                </textarea>
            </div>
            <div class="form-element">
                <button type="submit">Add New Comment</button>
            </div>
        </form>
    </div>

    <script src="http://code.jquery.com/jquery-latest.min.js"></script>
    <script type="text/javascript" src="script.js"></script>
</body>
</html>
```

4.) Agregamos un archivo llamado “script.js” con el siguiente contenido:

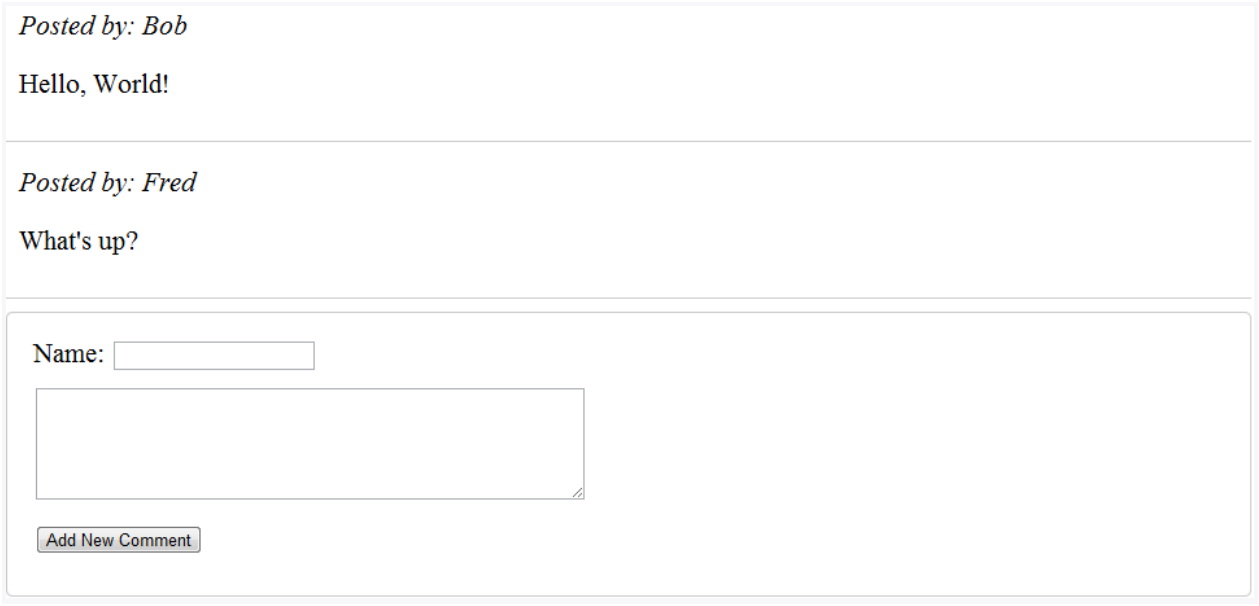
```
$(document).ready(function() {

    $('#comment-form').submit(function() {
        //Get the data from the form
        var name = $('#name').val();
        var comment = $('#comment').val();

        //Clear the form elements
        $('#name').val('');
        $('#comment').val('');

        addComment({
            name: name,
            comment: comment
        });
```

```
        return false;
    });

    function addComment(comment) {
        $('<div class="comment">')
            .append('<div class="author">Posted by: ' + comment.name + '</div>')
            .append('<p>' + comment.comment + '</p>')
            .appendTo('#comments');
    }

});
```

5.) Corremos la aplicación con el comando:

```
dpd --open
```

Se abrirá una pestaña en el navegador y se accederá a la página de la aplicación http://localhost:2403



Esta aplicación básica espera por un nombre y el cuerpo de un mensaje para postear un comentario.

Luego agregaremos un backend a esta aplicación, así el usuario puede interactuar con los demás y postear comentarios.

Creando el backend

1.) Abrimos el dashboard y tipeamos dashboard en la consola dpd.

1- Creamos una nueva colección llamada /comments
2- En las propiedades, agregamos una propiedad name y otra comment
3- En el editor, agregamos algunos comentarios de prueba

Integrandolo al frontend

En el archivo “index.html”, agregamos el siguiente script entre JQuery y script.js (línea 37):

```
<script type="text/javascript" src="/dpd.js"></script>
```

Esta línea generará una referencia a “dpd.js”, una librería simple dinámicamente creada para el backend. Detectará automáticamente qué recursos se agregaron a la aplicación, y los agrega al objeto dpd. Cada recurso tendrá funciones asíncronas para comunicarse con la aplicación Deployd.

En el script.js, agregar una función `loadComments()` dentro del evento $(document).ready:

```
function loadComments() {
    dpd.comments.get(function(comments, error) { //Use dpd.js to access the API
        $('#comments').empty(); //Empty the list
        comments.forEach(function(comment) { //Loop through the result
```

```
            addComment(comment); //Add it to the DOM.
        });
    });
}
```

Y llamarla cuando la página se termine de cargar:

```
$(document).ready(function() {

    loadComments();

    //...

});
```

Cuando se corra la aplicación, se deberá ver los comentarios creados en el dashboard

La función `get()` que hace que esto funcione envía una petición HTTP `GET` a `/comments`, y retorna un array de objetos en el recurso. No hay nada mágico en dpd.js; puedes usar peticiones estándar AJAX o HTTP si lo prefieres, o si eres incapaz de utilizar dpd.js. (ej. para aplicaciones móviles)

Nota: Si no has usado tanto AJAX, ten en cuenta que todas las funciones dpd.js son asíncronas y no retornan un valor directamente.

```
//No funciona:
var comments = dpd.comments.get();
```

Esto significa que tu JavaScript continuará ejecutándose y responderá al input del usuario mientras se cargan los datos, que hará que la app se sienta mucho más rápida para los usuarios.

Guardando datos

Observe que cualquier comentario que añada a través del formulario de la app desaparece al actualizar. Hagamos que el formulario guarde comentarios en la base de datos.

Elimina estas líneas del archivo `script.js` (cerca de la línea 10, dependiendo de donde pusiste la función `loadComments()`):

```
//Clear the form elements
$('#name').val('');
$('#comment').val('');

addComment({
    name: name,
    comment: comment
});
```

Y reemplazarlos con:

```
dpd.comments.post({
        name: name,
        comment: comment
}, function(comment, error) {
        if (error) return showError(error);

        addComment(comment);
        $('#name').val('');
        $('#comment').val('');
});
```

Agrega una función de utilidad en la parte superior de script.js para alertar cualquier error que obtenemos:

```
function showError(error) {
        var message = "An error occurred";
        if (error.message) {
```

```
                message = error.message;
        } else if (error.errors) {
                var errors = error.errors;
                message = "";
                Object.keys(errors).forEach(function(k) {
                        message += k + ": " + errors[k] + "\n";
                });
        }

        alert(message);
}
```

Un objeto `error` puede incluir ya sea una propiedad `message` o un objeto `errors` que contiene errores de validación.

Si cargas las página ahora, deberías ser capaz de cargar un comentario que aparece incluso después de actualizar la página.

7- Inconvenientes

Durante la instalación:

--Problema: No aparece el paquete node.js en los repositorios de Ubuntu 10.04

```
lucho9292@ubuntu:~$ sudo apt-get install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Couldn't find package nodejs
```

--**Solución:**

a.) Descargamos el paquete python-software-properties

```
lucho9292@ubuntu:~$ sudo apt-get install python-software-properties
```

b.) Agregamos el repositorio chris-lea/node.js

```
lucho9292@ubuntu:~$ sudo apt-add-repository ppa:chris-lea/node.js
gpg: keyring `/tmp/tmpVDZkwU/secring.gpg' created
gpg: keyring `/tmp/tmpVDZkwU/pubring.gpg' created
gpg: requesting key C7917B12 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpVDZkwU/trustdb.gpg: trustdb created
gpg: key C7917B12: public key "Launchpad chrislea" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
```

c.) Actualizamos la lista de paquetes de repositorios

```
lucho9292@ubuntu:~$ sudo apt-get update
```

--**Problema:**

Pueden obtenerse diversos errores si se trata de instalar deployd usando npm sin permisos de administrador

```
lucho9292@ubuntu:~$ npm install deployd -g
npm ERR! Error: EACCES, mkdir '/usr/lib/node_modules/deployd'
npm ERR!  { [Error: EACCES, mkdir '/usr/lib/node_modules/deployd']
npm ERR!   errno: 3,
npm ERR!   code: 'EACCES',
npm ERR!   path: '/usr/lib/node_modules/deployd',
npm ERR!   fstream_type: 'Directory',
npm ERR!   fstream_path: '/usr/lib/node_modules/deployd',
npm ERR!   fstream_class: 'DirWriter',
npm ERR!   fstream_stack:
npm ERR!    [ '/usr/lib/node_modules/npm/node_modules/fstream/lib/dir-writer.js:
36:23',
npm ERR!      '/usr/lib/node_modules/npm/node_modules/mkdirp/index.js:46:53',
npm ERR!      'Object.oncomplete (fs.js:107:15)' ] }
npm ERR!
npm ERR! Please try running this command again as root/Administrator.
```

--**Solución:**
Utilizar los permisos de usuario root para instalar Deployd

```
sudo npm install deployd -g
```

Durante el uso del programa

--Problema: Ocurrieron diversos problemas con respecto a los permisos de ejecución, en concreto el comando **`dpd --open`**,que sirve para correr la aplicación y abrir una pestaña del navegador. Era imposible correrlo si no se utilizaba el usuario root, lo cual traía complicaciones con el navegador Chromium ya que este no permite la ejecución como root.

--Solución: La solución fue utilizar el navegador Firefox, primero desinstalando Chromium y utilizando el comando **`sudo dpd --open`.**

8-Referencias

Deployd
Página Oficial: http://deployd.com/
Guia de Instalación: http://docs.deployd.com/docs/getting-started/installing-deployd.html

Node.js
Página Oficial: http://nodejs.org/
Wikipedia: https://es.wikipedia.org/wiki/Node.js
http://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/

NPM
Página Oficial: https://www.npmjs.com/
Wikipedia: https://en.wikipedia.org/wiki/Npm_%28software%29 (en ingles)

MongoDB
Pagina Oficial: http://www.mongodb.org/
Wikipedia: https://es.wikipedia.org/wiki/MongoDB