Node.js: how to check if a site has been added to Cloudflare

In this tutorial we will see how to check if a site has been successfully added to Cloudflare using Node.js.

A site successfully added to Cloudflare always has NS records set on Cloudflare's DNS servers and optionally an HTTP header with the value of this provider indicating the enabling of the proxy functionality.

First we will use the NPM module **got** (version 11.8.3 to avoid dynamic imports) to retrieve the site's HTTP headers. The NPM **validator** module will be used for class parameter validation.

```
'use strict';

const validator = require('validator');
const got = require('got'); // 11.8.3

class Request {
    constructor(url) {
        this.url = url;
    }

    async send() {
        if(!this.isValidURL(this.url) ) {
            return { error: 'Invalid parameter.' };
        }

        try {
            const response = await  got(this.url);
            return response.headers;
```

```
        } catch(err) {
            return { error: 'Request error.' };
        }
    }

    isValidURL(value) {
        if(typeof value !== 'string') {
            return false;
        }

        return validator.isURL(value);
    }
}
```

Now we need to retrieve the NS records array of the domain using the `Resolver` core object of the `dns` module which allows us to perform DNS queries using specific resolvers.

```
'use strict';

const { Resolver } = require('dns').promises;
const resolver = new Resolver();
const nameservers = ['8.8.8.8', '8.8.4.4'];
const validator = require('validator');

resolver.setServers(nameservers);

class DNS {
    constructor(host) {
        this.host = host;
        this.recordType = 'NS';
    }

    async resolve() {
```

```
        if(!this.isValidHost(this.host) ) {
            return { error: 'Invalid parameter.' };
        }

        try {
            return await  resolver.resolve(this.host,
this.recordType);
        } catch(err) {
            return { error: 'Record not found.' };
        }
    }

    isValidHost(value) {
        if(typeof value !== 'string') {
            return false;
        }

        return validator.isFQDN(value);
    }
}
```

We just have to create the main class that will use the two helper classes and add two methods which will check for the presence of the Cloudflare NS records and the HTTP header with the specified value.

```
class Cloudflare {
    constructor(host) {
        this.host = host;
        this.dns = new DNS(this.host);
        this.request = new
Request(`https://${this.host}`);
    }

    hasHostNSRecords(records) {
```

```
        if(!Array.isArray(records) || records.length
=== 0) {
            return false;
        }
        return records.every(record => { return
record.includes('ns.cloudflare.com'); });
    }

    hasHostHTTPHeader(headers) {
        if(typeof headers !== 'object' || headers ===
null) {
            return false;
        }
        return typeof headers.server !== 'undefined'
&& headers.server === 'cloudflare';
    }

    async check() {
        try {
            const records = await this.dns.resolve();
            const headers = await
this.request.send();
            return this.hasHostNSRecords(records) ||
this.hasHostHTTPHeader(headers);
        } catch(err) {
            return false;
        }
    }
}

module.exports = Cloudflare;
```

Example of use:

```
'use strict';

const Cloudflare = require('./lib/Cloudflare');
const cloudFlare = new
Cloudflare('gabrieleromanato.dev');

(async () => {
    try {
        console.log(await cloudFlare.check());
    } catch(err) {
        console.log(err);
    }
})();
```