



# Installation du serveur Node.js et de l'application Ether

---

Dans ce bref tutoriel, nous allons vous présenter comment installer sur votre machine (et faire marcher) notre application ETHER, ainsi que le serveur associé, Node.js.

## I. Installation du serveur et de l'application :

### *1. Méthode simple et rapide :*

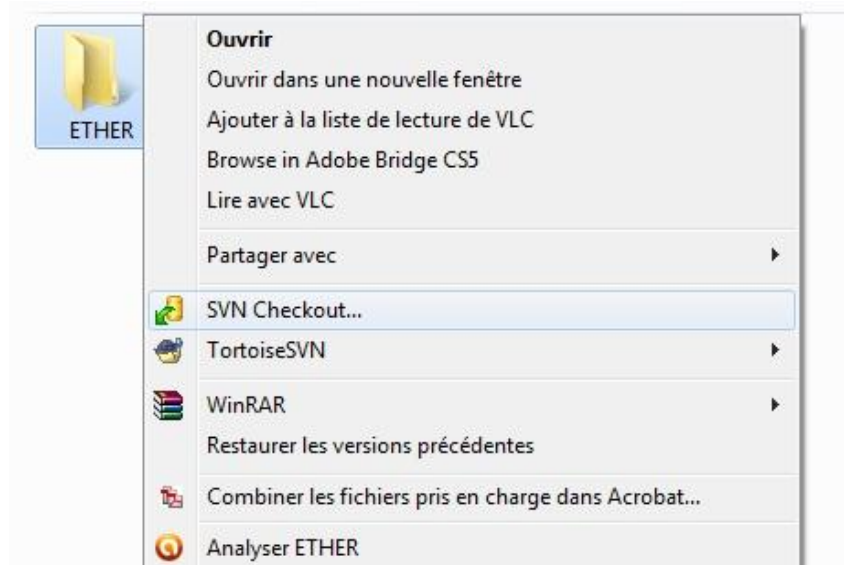
La méthode la plus simple consiste à se rendre sur la page de téléchargement de l'espace « Google Code » dédié au projet Ether, et à y télécharger les deux archives « node\_js.zip » et « projet\_ether.zip ». L'adresse de la page web est la suivante : <http://code.google.com/p/projet-ether/downloads/list>.

Après téléchargement des deux archives, il suffit de les dézipper dans un répertoire commun (de façon à placer le serveur node.js dans le même dossier que le fichier « app.js »), disons par exemple « C:\ETHER\ » et le tour est joué !

### *2. Méthode plus longue mais plus complète :*

La seconde méthode consiste à télécharger l'ensemble des fichiers de notre dépôt SVN sur « Google Code ». Pour cela, il est nécessaire de télécharger un client Subversion, tel que [TortoiseSVN](#) ou autre (une liste de clients SVN est disponible sur le [site d'Apache](#)).

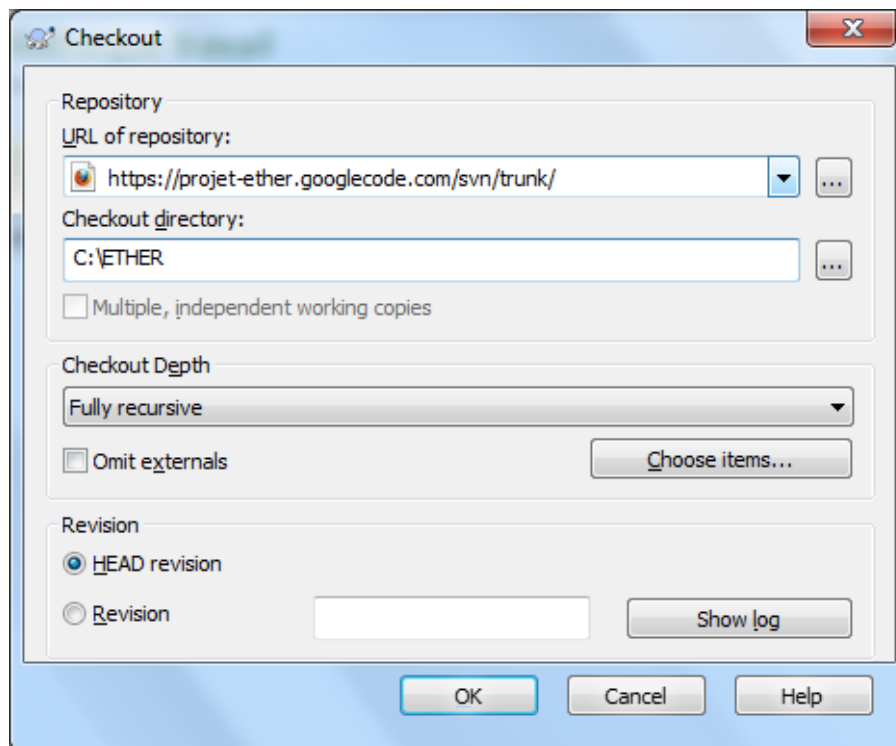
Une fois le client SVN installé, il faut alors créer un dossier vide, disons par exemple « C:\ETHER\ » puis effectuer un « Checkout » dans ce dossier. Si vous utilisez TortoiseSVN, il vous suffit de faire un clic droit sur le dossier, et de sélectionner l'option « SVN checkout... »



Une fenêtre apparaît alors et propose de sélectionner l'adresse du dépôt SVN. Cette adresse est accessible depuis [la page « source »](https://projet-ether.googlecode.com/svn/trunk/) de l'espace « Google Code » de notre projet. Il s'agit de :

**`https://projet-ether.googlecode.com/svn/trunk/`**

Une fois l'adresse copiée dans le premier cadre, il suffit de vérifier que le dossier prévu pour le Checkout est bien le bon et que les options sont correctement cochées (à savoir « Checkout fully recursive » et « Head Revision ») avant de lancer le Checkout.



Il est alors nécessaire d'être un peu patient car notre dépôt comporte un nombre important de fichiers (environ 20 000) mais une fois le CheckOut fini, tout ce dont nous avons besoin a été téléchargé.

### *3. Méthode plus compliquée mais plus consciencieuse :*

La dernière méthode, consiste à installer le serveur Node.js ainsi que ses plug-ins de manière indépendante, en le téléchargeant directement sur [le site officiel](#). Une fois cette première étape passée, il faut alors passer à l'installation des modules complémentaires. Pour cela, il est nécessaire d'ouvrir une fenêtre de commande (dans Windows, taper « cmd » dans le menu Démarrer).

Grâce à l'installation de Node.js, deux commandes sont désormais accessibles : « node » et « npm ». Pour installer les deux plug-ins dont nous avons besoin, nous allons faire appel à npm. Les deux lignes de commande à taper sont donc les suivantes :

```
npm install express  
npm install socket.io
```

Une fois l'installation du serveur complétée, vous pouvez maintenant télécharger l'application Ether en suivant l'une des deux premières méthodes présentées dans ce tutoriel.

Vous pouvez aussi avoir envie de mettre à jour la version de Dojo utilisée par ETHER, à savoir la version 1.7.2, la plus récente disponible à ce jour (25 mars 2012). Pour cela vous pouvez télécharger le framework Dojo sur [son site officiel](#). Cependant, il est déconseillé de le faire car nous utilisons dans notre application une « Custom Build » de Dojo puisque nous avons écrit nous-mêmes quelques modules complémentaires, et il faudra donc les inclure dans la nouvelle version pour qu'ETHER continue de fonctionner.

## **II. Lancement de l'application :**

Une fois l'installation terminée, il faut maintenant démarrer le serveur afin de pouvoir accéder à l'application. Pour cela, il faut ouvrir une fenêtre de commande (« cmd » dans Windows). Attention, il ne faut pas qu'un serveur Apache soit déjà lancé sur la machine en question, sinon il se produira un conflit puisque les deux serveurs utilisent le même port (80).

Si la première méthode a été utilisée, le script « Node.js » se trouve dans le dossier « C:\ETHER », il faut donc s'y placer puis lancer le serveur grâce aux deux lignes de commande suivantes :

```
cd "C:/ETHER"  
node app.js
```

Si la seconde méthode a été préférée, tous les fichiers présents sur le dépôt ont été téléchargés. Il est donc nécessaire de se placer dans le dossier « deployment » qui contient la version finale de notre projet. Les deux lignes de commande sont donc les suivantes :

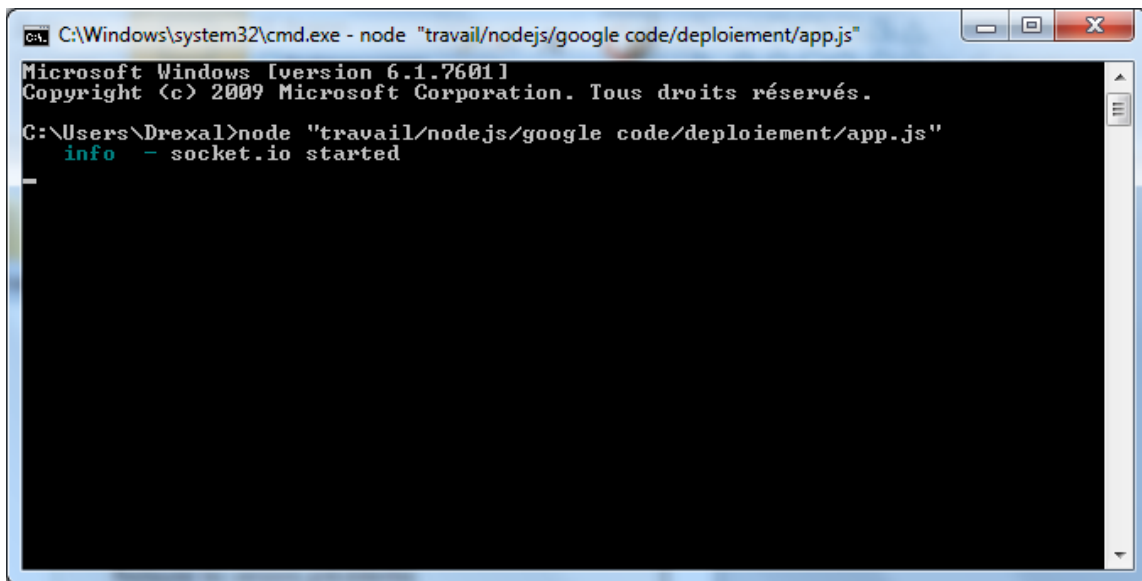
```
cd "C:/ETHER/deploiement"

node app.js
```

Et enfin si c'est la troisième méthode qui a été choisie, la commande « node » est accessible depuis n'importe quel répertoire. Une seule ligne de commande suffit donc, mais il ne faut pas oublier de préciser le chemin absolu (et non pas relatif) du script « app.js ». Cela devrait donc ressembler à :

```
node "C:/ETHER/app.js"
```

Lorsque le serveur démarre, un message de confirmation s'affiche dans la console : « socket.io started » comme on peut le voir ci-dessous :



```
C:\Windows\system32\cmd.exe - node "travail/nodejs/google code/deploiement/app.js"
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.
C:\Users\Drexal>node "travail/nodejs/google code/deploiement/app.js"
info - socket.io started
```

A ce moment, le plus dur est fait ! Pour se connecter à ETHER, il suffit maintenant d'ouvrir un navigateur internet suffisamment récent (et différent d'Internet Explorer) et de taper dans la barre d'adresse « http://localhost »

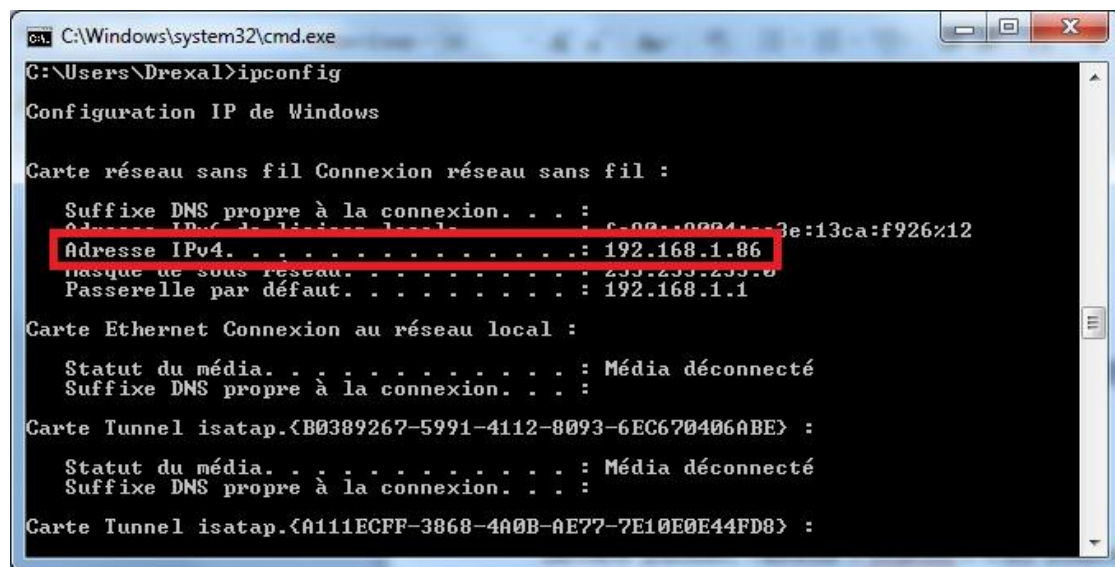


## Connexion à ETHER

☒ Participant ☐ Animateur

Dernière précision, l'adresse « localhost » n'est évidemment accessible que sur la machine sur laquelle le serveur Node.js a été lancé. Pour connecter d'autres périphériques (ordinateurs, smartphones, tablettes, etc...) il est nécessaire que ces derniers soient connectés sur le même réseau local (WiFi par exemple) que celui sur lequel se trouve la machine qui héberge le serveur Node.js.

Si les périphériques se trouvent sur le même réseau, alors il suffit qu'ils se connectent directement à l'adresse IP de la machine qui héberge le serveur Node.js. Pour connaître cette adresse, il suffit de lancer la commande « ipconfig » dans la fenêtre de commande (sous Windows en tout cas...).



```
C:\Windows\system32\cmd.exe
C:\Users\Drexal>ipconfig

Configuration IP de Windows

Carte réseau sans fil Connexion réseau sans fil :
    Suffixe DNS propre à la connexion. . . : 
    Adresse IPv6 de liaison locale. . . . : fe80::8284::3e:13ca:f926%12
    Adresse IPv4. . . . . : 192.168.1.86
    Masque de sous-réseau. . . . . : 255.255.255.0
    Passerelle par défaut. . . . . : 192.168.1.1

Carte Ethernet Connexion au réseau local :
    Statut du média. . . . . : Média déconnecté
    Suffixe DNS propre à la connexion. . . : 

Carte Tunnel isatap.{B0389267-5991-4112-8093-6EC670406ABE} :
    Statut du média. . . . . : Média déconnecté
    Suffixe DNS propre à la connexion. . . : 

Carte Tunnel isatap.{A111ECFF-3868-4A0B-AE77-7E10E0E44FD8} :
```

Enfin, lorsque la session de travail prend fin, il est possible d'éteindre le serveur Node.js en appuyant sur « ctrl + C » dans la fenêtre de commande.

Remarque : De nombreuses informations concernant le fonctionnement de l'application, telles que la connexion d'un nouveau participant, sa déconnexion, ou encore l'envoi de messages, s'affichent dans la console du serveur. Elles n'ont aucune importance mais peuvent être utiles en cas de crash du serveur pour identifier le problème.