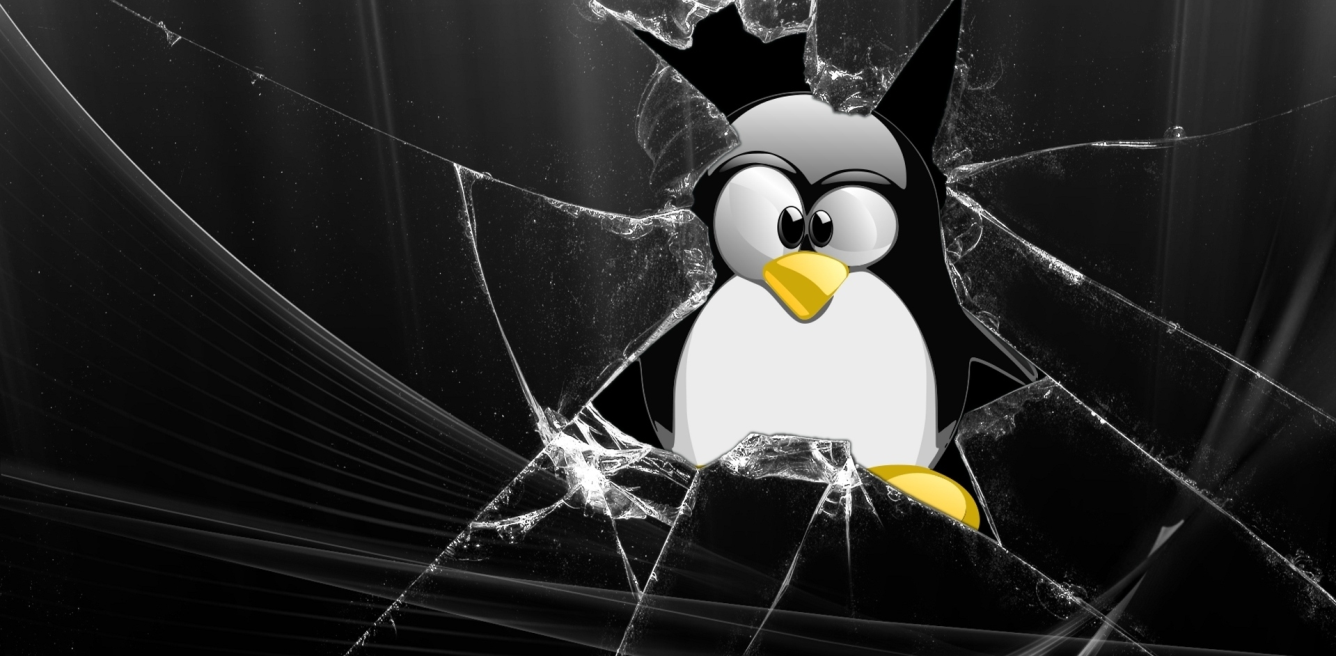
**Use git, ssh and npm on windows with Git Bash**

[**24 January 2017**](https://www.theodo.fr/blog/2017/01/use-git-ssh-and-npm-on-windows-with-git-bash/)[**1 Comment**](https://www.theodo.fr/blog/2017/01/use-git-ssh-and-npm-on-windows-with-git-bash/#disqus_thread)[**Nicolas Ngô-Maï**](https://www.theodo.fr/blog/author/nicolasntheodo-fr/)**3 min read**



If you are used to develop on Linux and you have to suddenly switch back to Windows for a particular project, it can be really painful to use native tools like putty or power shell to develop.  
Don’t worry, there are plenty of solutions to make things right.  
For example, you could work on a Linux virtual machine inside your Windows.  
Another solution (the one I chose) is to use Git Bash.  
This setup has several advantages:

* It requires no installation, which means it can be set up without admin rights
* It is rather lightweight and easily packageable
* You still have access to your windows filesystem via your command line

We are going to configure and package nodejs/npm inside the Git Bash to share it with every developers of your new team.

**Get the softwares**

First, get the desired softwares and add these to C:\Applications\ (where you will have sufficient rights to execute them):

* Download Git for Windows Portable (“thumbdrive edition”) [**> HERE <**](https://git-scm.com/download/win) and install it (ex: in C:\Applications\git)
* Download node with npm zip package [**> HERE <**](https://nodejs.org/en/download/releases/) and unzip it (ex: in C:\Applications\node)

Git portable edition comes with Git Bash included.

You should now have a folder which looks like this :

**Applications**

├─ **git**

│ ├─ **git\_bash**.exe

│ └─ **etc**

│ ├─ **profile** (**edited**)

│ └─ **node\_env**.conf

└─ **node**

├─ **node**

└─ **npm**

**Configure your shell**

In git Bash, a Linux like environement is simulated so you can access your Linux filesystem with Linux style paths.  
For example, to print the content of C:\Applications\ you can type:

ls /c/Applications

Open git bash and type the following command:

export PATH=$PATH:/c/Applications/node

Type then node --version and npm --version to check that node and npm are available.  
If not, check that you have the npm and node.exe files in C:\Applications\node.

To reproduce this configuration when launching your terminal, you can create a custom configuration file in C:\Applications\git\etc\node\_env.conf :

*# Include node PATH*

PATH=$PATH:/c/Applications/node

Edit your etc/profile git bash file (Windows location: C:\Applications\git\etc\profile) and add the following line just before exporting the PATH:

source "etc/node\_env.conf"

You should then be able to use npm and node out of the box!  
For example, if you want to contribute to this [**planning burndown chart project**](https://github.com/theodo/planning-bdc):

git clone https://github.com/theodo/planning-bdc.git && cd planning-bdc

npm install

npm run watch

And here you go!

**Use SSH in your shell**

And now how about using SSH to connect to github?  
Indeed, Git Bash comes with a lot of features from Linux like grep, find, sed, and even ssh and scp!  
To create a SSH key:

**ssh-keygen** **-t** **rsa** **-b** 4096 **-C** "**your\_email**@**example**.**com**"

As in Linux, the key file location will be in ~/.ssh:

cat $HOME/.ssh/id\_rsa.pub

Add it to Github and run:

git clone git@github.com:theodo/planning-bdc.git && cd planning-bdc

npm install

npm run watch

And here you go again!

**Set up your proxies (optional)**

If you get stuck when using git clone or npm install, you may be blocked by a proxy (which requires configuration as well). Find out what the address is following [**this process**](http://superuser.com/questions/346372/how-do-i-know-what-proxy-server-im-using).

Add the following lines to your node\_env.conf :

**if** [ -f "etc/node\_env.var" ];

**then**

source "etc/node\_env.var"

**else**

touch etc/node\_env.var

read -p "What is your proxy login? " PROXY\_LOGIN

echo "PROXY\_LOGIN=\"${PROXY\_LOGIN}\"" >> etc/node\_env.var

read -p "What is your proxy password? " PROXY\_PASSWORD

echo "PROXY\_PASSWORD=\"${PROXY\_PASSWORD}\"" >> etc/node\_env.var

PROXY\_ADDRESS="proxy.address.com"

echo "PROXY\_ADDRESS=\"${PROXY\_ADDRESS}\"" >> etc/node\_env.var

PROXY\_PORT="8080"

echo "PROXY\_PORT=\"${PROXY\_PORT}\"" >> etc/node\_env.var

**fi**

*# Used for Node Terminal proxy (for packages as shipit)*

export HTTP\_PROXY=http://${PROXY\_LOGIN}:${PROXY\_PASSWORD}@${PROXY\_ADDRESS}:${PROXY\_PORT}

export HTTPS\_PROXY=https://${PROXY\_LOGIN}:${PROXY\_PASSWORD}@${PROXY\_ADDRESS}:${PROXY\_PORT}

*# Npm proxy*

npm config set proxy $HTTP\_PROXY

npm config set https-proxy $HTTPS\_PROXY

*# Git proxy*

git config --global http.proxy $HTTP\_PROXY

git config --global https.proxy $HTTPS\_PROXY

This piece of bash code will :

* Check if a node\_env.var file exists
* If true, it will import it
* If false, it will create it and fill it with the PROXY\_LOGIN, PROXY\_PASSWORD, PROXY\_ADDRESS and PROXY\_PORT variables
* Then it will configure your node, npm and git with the proxy

**Be careful with your HTTPS proxy, it might have a different address / port than your http proxy**

Start your Git Bash and you should be asked for the proxy variables. You should now be able to use git clone and npm install for your applications!

You can also use the integrated ssh client.

If you need to change the variables, destroy the node\_env.var file and you will be asked for new values.

**Conclusion**

Now you got a configured and packaged Git Bash, you can adapt it to create your own environement!  
Thanks for reading, don’t hesitate to leave feedback!

You liked this article? You'd probably be a good match for our ever-growing tech team at Theodo.