

# Contributing code to PrestaShop -

## Table of contents


- [Contributing code to PrestaShop](#)
  - [If you already know how to work with Git/GitHub](#)
  - [Working using Git](#)
    - [Complete process to your first pull request](#)
      - [1. Creating a local clone of the PrestaShop repository](#)
      - [2. Making your changes and pushing them to your GitHub repository](#)
      - [3. Submitting your changes to the PrestaShop developers](#)
    - [Keeping your fork and clone up to date](#)
    - [Making further pull requests for PrestaShop](#)
    - [Making a pull request for one of PrestaShop's modules](#)
    - [Understanding Git](#)
  - [Making changes directly in GitHub](#)

## Contributing code to PrestaShop

As an open source project, PrestaShop thrives thanks to its community, and especially from volunteer developers who are willing to give some of their time to help resolve issues in the code.

There are several ways to approach this:

- If you find an issue but do not know how to write code for it, please report it on the PrestaShop Forge, our bug-tracking system: read the "[How to use the Forge to contribute to PrestaShop](#)" chapter to learn more.
- If you are a software developer, you can help the development of PrestaShop by fixing problems that have been reported by you or another person in [the Forge](#).
  - If you want to help fix a bug by writing a patch, there are a few steps to follow. In short, your submission should be made as a [pull request on GitHub](#). See below.
  - If you have never used Git or GitHub, either read about these two by clicking on the links in the "Understanding Git" section of this page, or learn how to edit a single file within GitHub by reading the "Making changes directly in GitHub" section of this page.

 As a contributor, you retain the copyright to your code. However, by submitting it to PrestaShop's repository, you are releasing it under either of these licenses:

- For general PrestaShop files: the [OSL 3.0 license](#).
- For module files: the [AFL 3.0 license](#).

Most of the instructions on this page will require you to [create your own GitHub account](#).

If you already know how to work with Git/GitHub

Then you're all set! Please work exclusively with pull requests.

Here are technical information:

- Contributing to PrestaShop Core:
  - Use this repository: <https://github.com/PrestaShop/PrestaShop>
  - Branches to use for your pull requests:
    - **'develop'** for the forthcoming PrestaShop 1.7: new features, major changes, etc.
    - **'1.6.1.x'** for the current PrestaShop 1.6: bugfixes and security fixes only, NO NEW FEATURE, even small ones. The PR's merger will take care of porting your fix to the 'develop' branch
- Contributing to a native module:
  - Each native module has its own repository. For instance, <https://github.com/PrestaShop/autoupgrade>
  - Branches to use:
    - **'dev'** for everything: fixes, new features, etc.
    - **'master'** is only used by the Core devs for the released version.

A few things to keep in mind:

- If you are working on more than a simple bugfix, please [open a Forge ticket](#) about so that the Product team knows about it. See [How to use the Forge to contribute to PrestaShop](#).
- Use the [PSR-2 coding style guide](#). See [Coding Standards](#).
- Follow the guidelines for [writing a commit message](#).
- Do not use GitHub issues: let us know about bugs and ideas [through the Forge](#).

[Read the Build devblog](#) regularly in order to get the latest news!

## Working using Git

### **This is the recommended way of contributing code to PrestaShop Core.**

In order to make changes to PrestaShop's core code, you first need to create a local fork of PrestaShop, or at least have your own GitHub repository. When your changes are done, you must send your work to the PrestaShop developers for approval. This means installing a tool to retrieve the PrestaShop code from GitHub, editing it on your machine, and sending your changes back to PrestaShop's GitHub repository.

If all this sounds like a foreign language, you should learn about Git and GitHub first. Go to the "Understanding Git" section of this page.

### **Complete process to your first pull request**

In the process presented herein, we will work on Windows and use the TortoiseGit tool. Be sure to install it before you start. You can download it here: <https://tortoisegit.org/>.

There are many more available GUI tools for Git on both Windows and OS X: <http://git-scm.com/downloads/guis>. Windows users should also install the Git for Windows extension: <https://git-for-windows.github.io/> (with the "Git bash here" option enabled).

If you are using a Unix system, we figure you should be happier with [Git's own command line tool](#) 🍷. Read all about it here: <http://git-scm.com/book>.

#### **i Cloning with the modules**

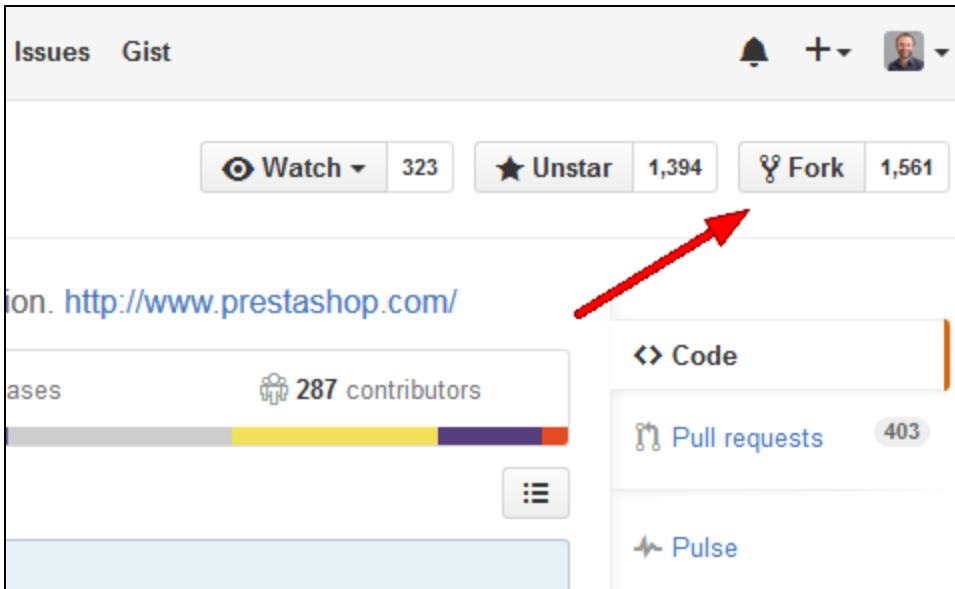
Since March 20th, 2014, all of PrestaShop's default modules have been moved into their own GitHub repositories (or "submodules"). This implies a few changes on your side:

- You are starting afresh:
  - You are cloning the project: add the `--recursive` option in order to also clone all the module's repositories (or check the "Recursive" box in TortoiseGit). See below for more about the process.
- You already have a local clone:
  - You want to update your cloned project: use `git pull && git submodule update --init --recursive`
  - You want to switch the submodules on the master branch: use `git submodule foreach git checkout master`
  - You want to retrieve the submodules log files: use `git submodule foreach git log`

### **1. Creating a local clone of the PrestaShop repository**

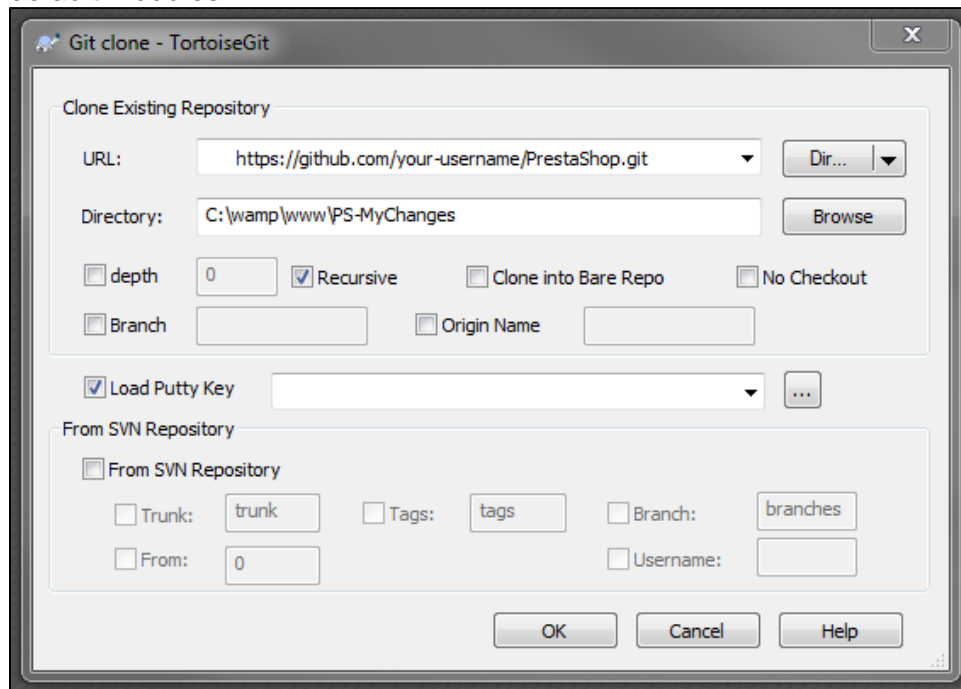
Connect to your GitHub account, and then:

1. Go to the [PrestaShop GitHub repository](#).
2. Fork the project by clicking on the "Fork" button. The fork will appear as a "PrestaShop" repository in your own account, with all the original branches.



You've successfully forked the PrestaShop repository, but so far it only exists on your GitHub account. To be able to properly work on the project, you will need to clone it to your local machine.

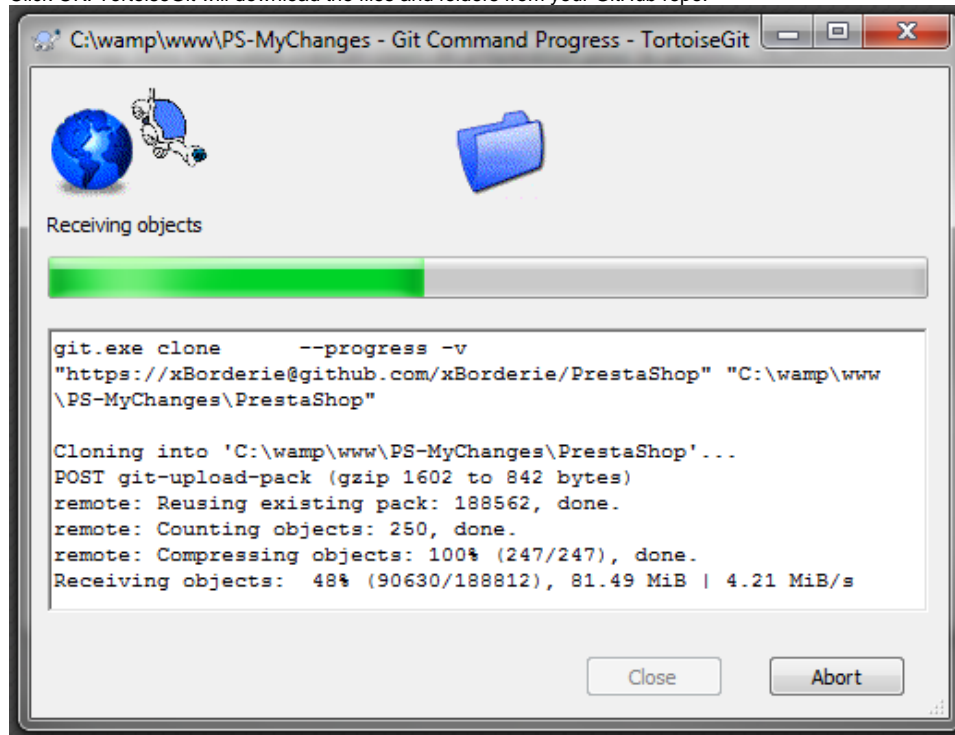
3. On your local machine:
  - a. Clone your fork:
    - i. Create a new folder on your local machine, preferably in your local webserver's public file folder ('www', 'htdocs', or something else depending on our installation). We'll name it "PS-MyChanges", but you can use whatever name you want.
    - ii. Get the HTTPS clone URL from your GitHub PrestaShop repository. It should resemble this: <https://github.com/your-username/PrestaShop.git>
    - iii. Right-click on your PS-MyChanges folder, and in the Git section of the menu that appears, choose "Git Clone..."
    - iv. In the "Git clone" window that opens, paste the clone URL in the "URL" field and remove the final "\PrestaShop" mention in the "Directory" field, so that the files are downloaded at the root of the folder.  
Finally, do not forget to check the "Recursive" box so that you will also get all the default modules!



⚠ If you forgot to tick the "Recursive" checkbox when cloning your fork, use a command line terminal (shell, Bash, etc.) and enter these commands:

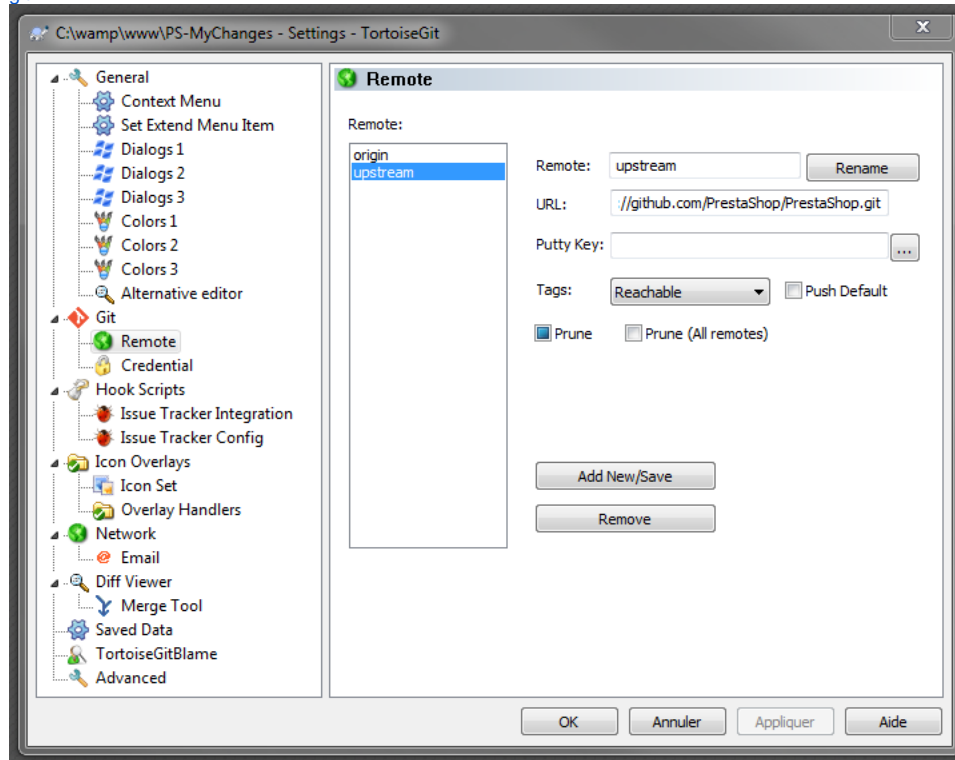
```
git submodule init
git submodule update
```

- If you want TortoiseGit to remember your GitHub username instead of asking it every time you push content to your online repository, you can add that username to your remote URL. For instance, if your username is your-username: <https://your-username@github.com/your-username/PrestaShop.git>. Note that TortoiseGit will still ask for your password each time you push changes to your GitHub account. Therefore, you can also store your password in the remote URL, but it is not recommended, for obvious reasons. Here is how: <https://your-username:MyPaSsWoRd@github.com/your-username/PrestaShop.git>.
- v. Click OK: TortoiseGit will download the files and folders from your GitHub repo.



- b. *(optional)* Add the "upstream" remote: When a repository is cloned, it has a default remote repository called "origin" that should point to your fork on GitHub, not to the original PrestaShop repository from which it was forked. To keep track of the changes in the original repository and keep your local clone updated, you need to add another remote URL named "upstream":
- i. Right-click on the PrestaShop folder in the PS-MyChanges folder (it should have a green mark if no change has been made yet), and at the bottom of the TortoiseGit submenu, choose "Settings". This opens the Settings panel.
  - ii. Open the Git > Remote section of the Settings panel.
  - iii. Remote field: Type "upstream".

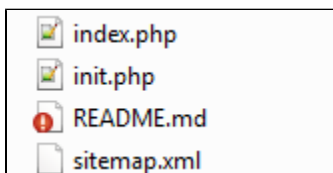
- iv. URL field: put the HTTPS clone URL from the official PrestaShop GitHub repository: <https://github.com/PrestaShop/PrestaShop.git>.



- v. Click the "Add new/Save" button. TortoiseGit asks you if you want to disable tag fetching for this remote: choose "No", since this remote IS an official remote.
- vi. TortoiseGit asks you if you want to fetch remote branches from the newly added remote: choose "Yes".
- vii. TortoiseGit displays the Fetch window for PS-MyChanges: click "OK". A window will open showing that Git is fetching data. Close it once it is done, and close the still-open Settings window (which should by now list "upstream" below "origin" in the list of remote URLs).

## 2. Making your changes and pushing them to your GitHub repository

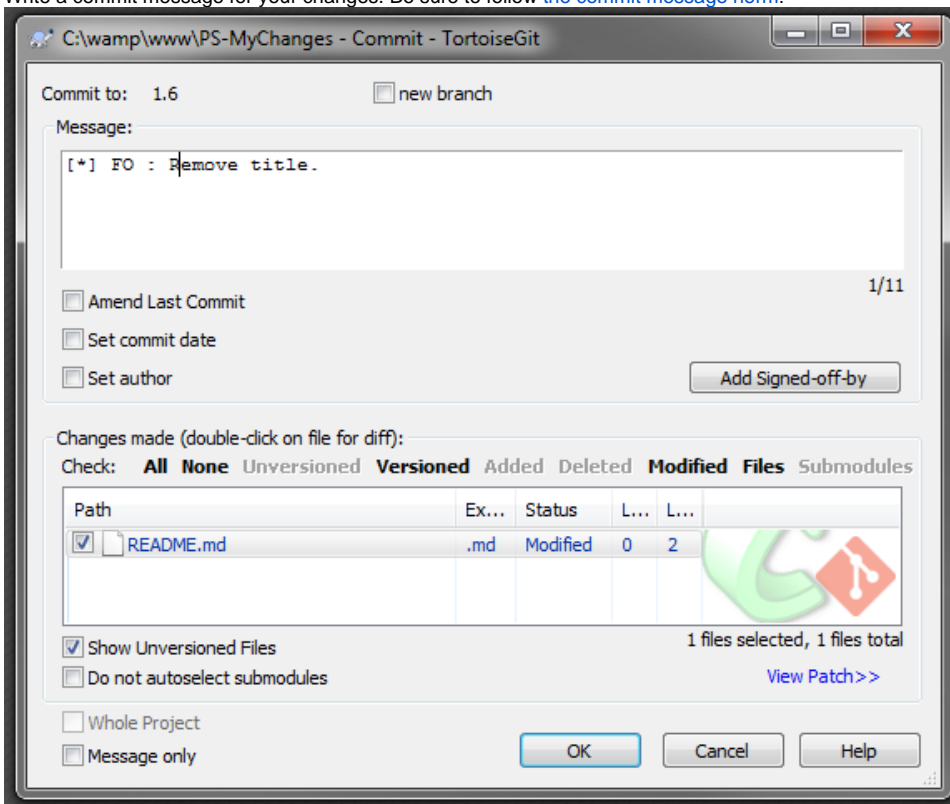
Edit the files that you want to change. As soon as you save an edited file, Windows will change the file's icon from green to red to indicate that it has been changed since the last update.



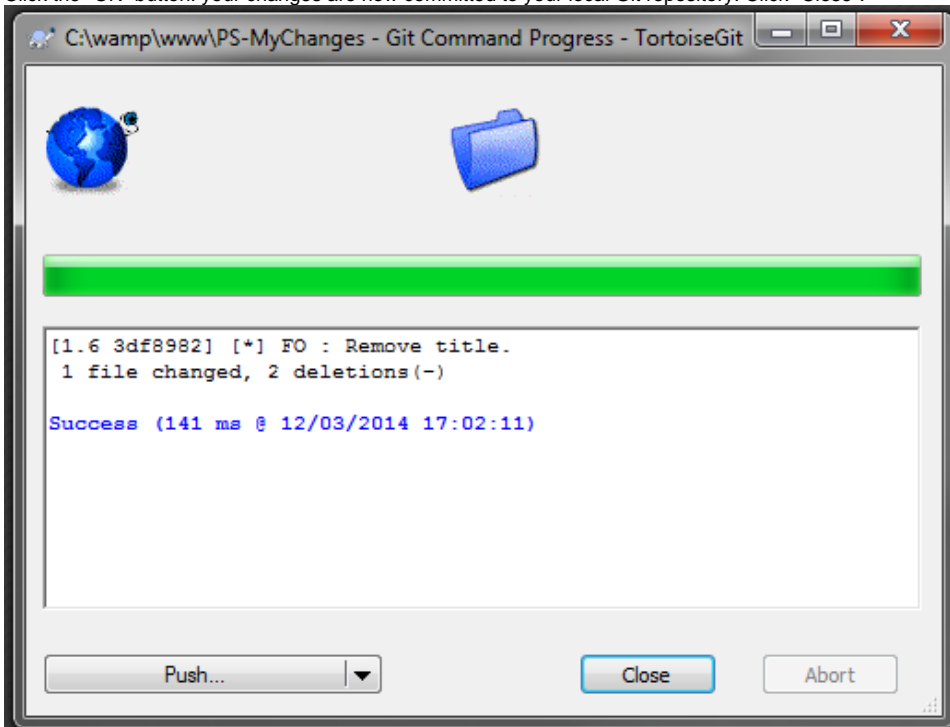
Once you are done editing the local files:

1. Reach the root of your local repository (the folder above PS-MyChanges). Right-click on the PS-MyChanges and choose "Git Commit -> develop...", with "develop" being your current local branch. This will open the Commit window. The files you have changed should appear in the "Changes made" section.
2.
  - Make sure to only commit the files that you meant to change for this feature/bugfix, but not any other changed files! For instance, PrestaShop and its modules might change some configuration files: do not ever commit them! If there are files that you did not change yourself or do not want to commit at this time, simply deselect them before clicking "OK".

3. Write a commit message for your changes. Be sure to follow [the commit message norm!](#)

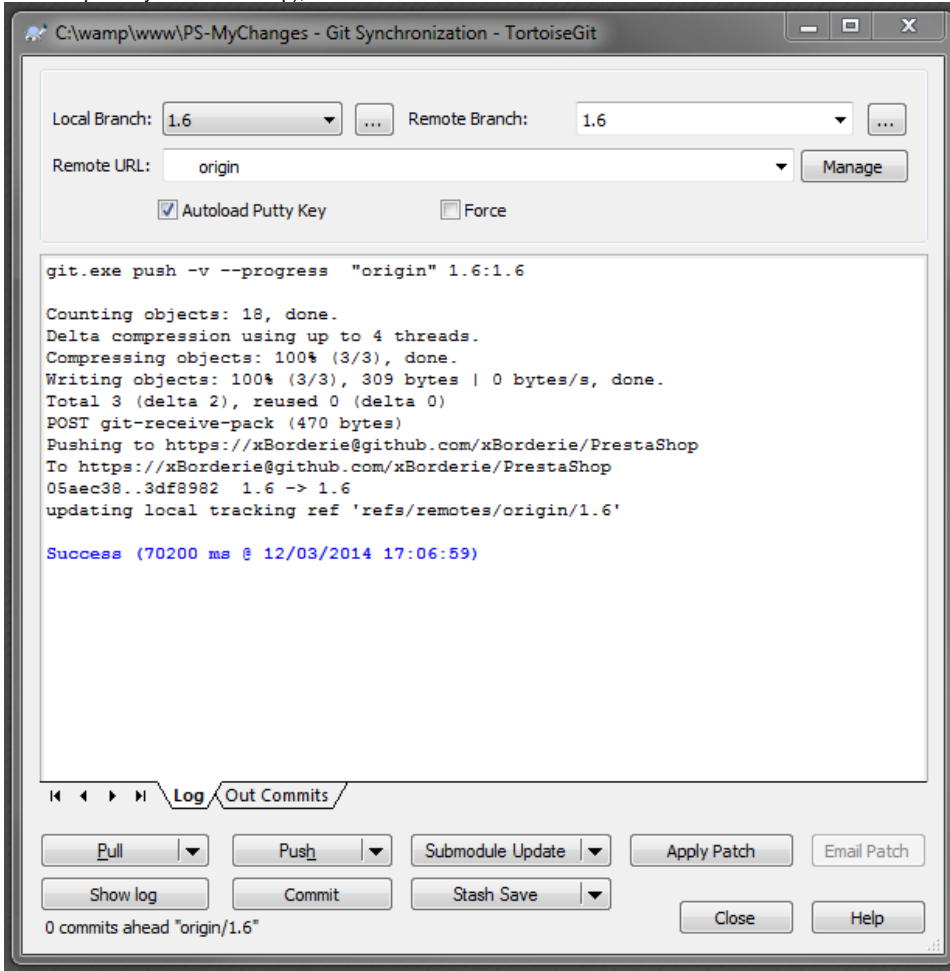


4. Click the "OK" button: your changes are now committed to your local Git repository. Click "Close".



5. Right-click on the PS-MyChanges folder and choose "Git Sync...": the Git Synchronization window opens, asking you which local branch should be pushed to which remote branch – usually it should be "develop" in both cases, unless you have created a specific local branch your set of

changes. Leave "origin" as the remote URL since you can only push your changes to your own repository on GitHub (and not the "upstream" official repository for PrestaShop), and click "Push".



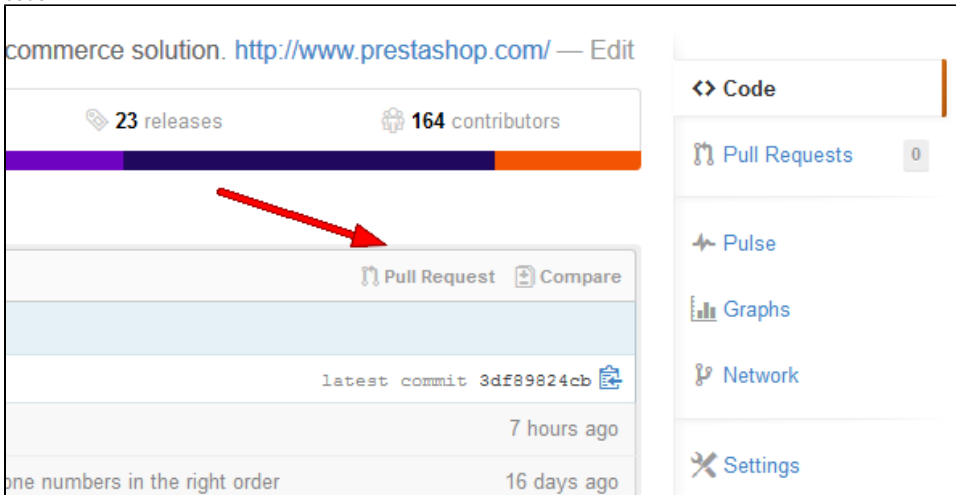
6. TortoiseGit updates the window with the current status, and might also open an OpenSSH window asking you for your GitHub username, then your GitHub password. Give them both and validate: the status window should push your changes to your GitHub repository.
7. Success! You can close the TortoiseGit windows: the rest of the process will happen on GitHub.

### 3. Submitting your changes to the PrestaShop developers

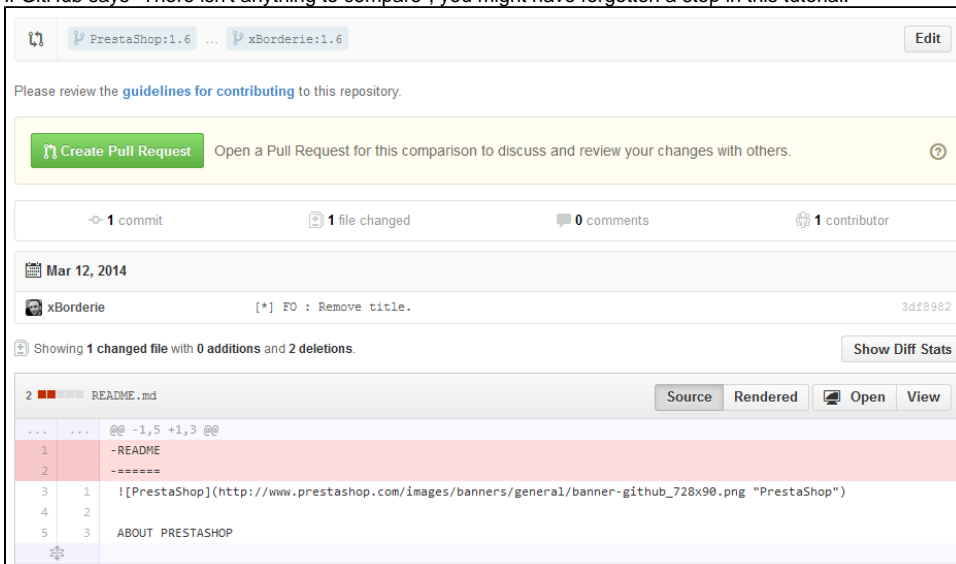
Now that your changes are stored on your GitHub fork, you need to let them be known to the PrestaShop developers. This is done by creating a "pull request" (or PR), which can be seen as making a code suggestion using all the files you have committed. GitHub has an excellent [help page about creating pull requests](#), but here is a summary:

1. Go to your own PrestaShop repository on GitHub.

2. Either click on the "Pull Request" button or the "Compare" button. Both will open the same screen, comparing your latest commit to the original code.



3. The top of the screen indicates what is being compared. It should be PrestaShop's "develop" branch with your own "develop" branch. You should not have anything to change here: just make sure that the base fork is the PrestaShop repository at branch "develop", and that the head fork is your own repository at the branch you pushed your changes to. If GitHub says "There isn't anything to compare", you might have forgotten a step in this tutorial.



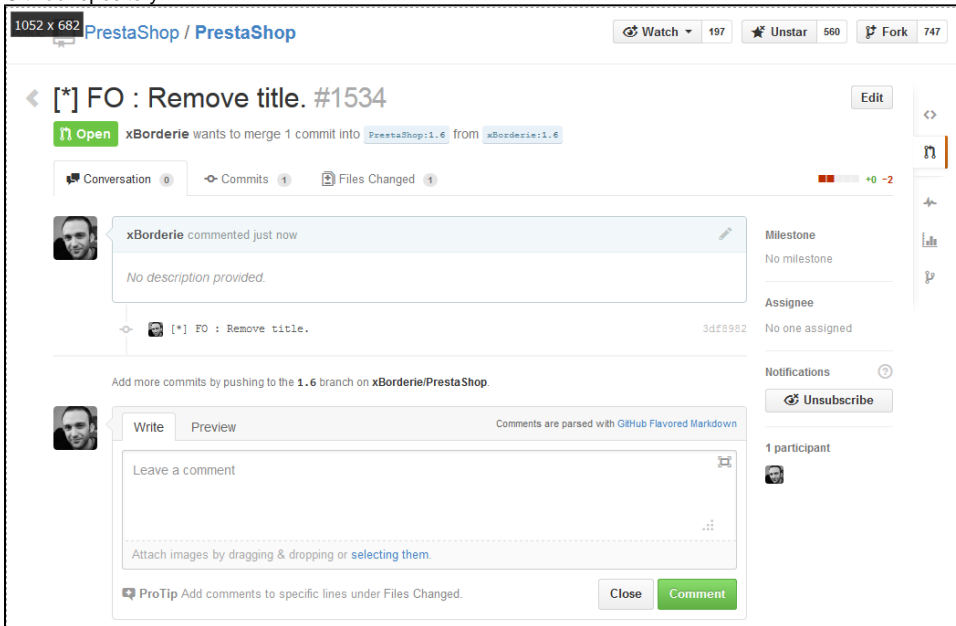
4. Click on the "Create a pull request" button. A new screen should appear, in which you can review your changes and give your pull request a title (it should already use the commit message you used from your local clone).



- This title is only for internal use at GitHub: it helps the PrestaShop developers, but only your local commit message is kept in the project's Git history. In any case, most of the time you won't need to edit the title: keep the commit message.

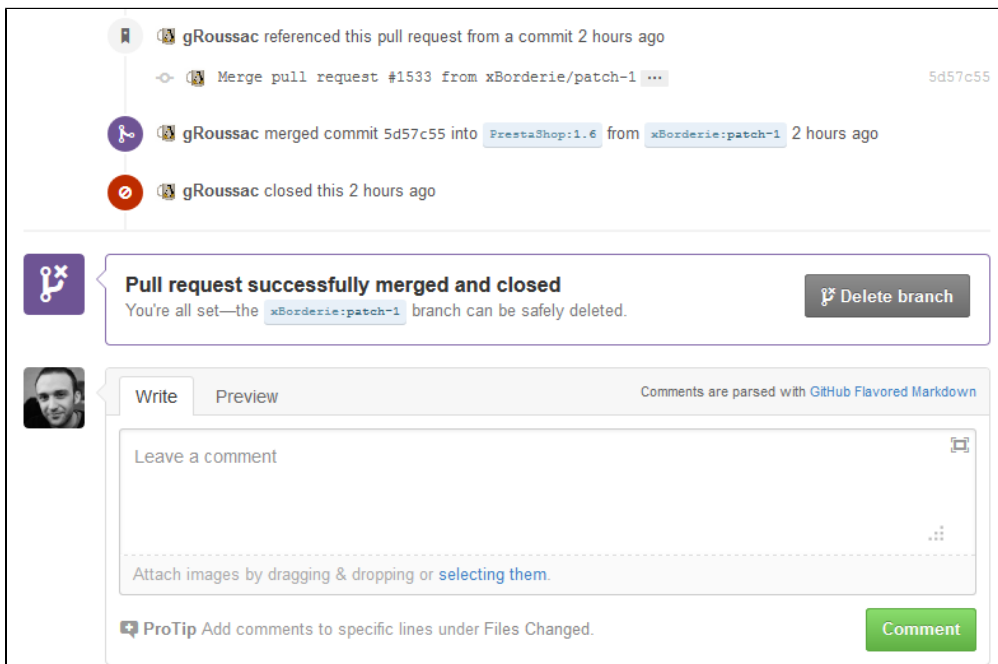


5. Click on the "Send pull request" button: GitHub opens a screen where you can start a conversation about your pull request, right in PrestaShop's GitHub repository.



6. Congratulations, your changes are now ready to add to the PrestaShop codebase!

Wait for one of the core developers either to include your change in the codebase, or to comment on possible improvements you should make to your code. Once your pull request is accepted, you will receive an e-mail, and the discussion section will look like this:

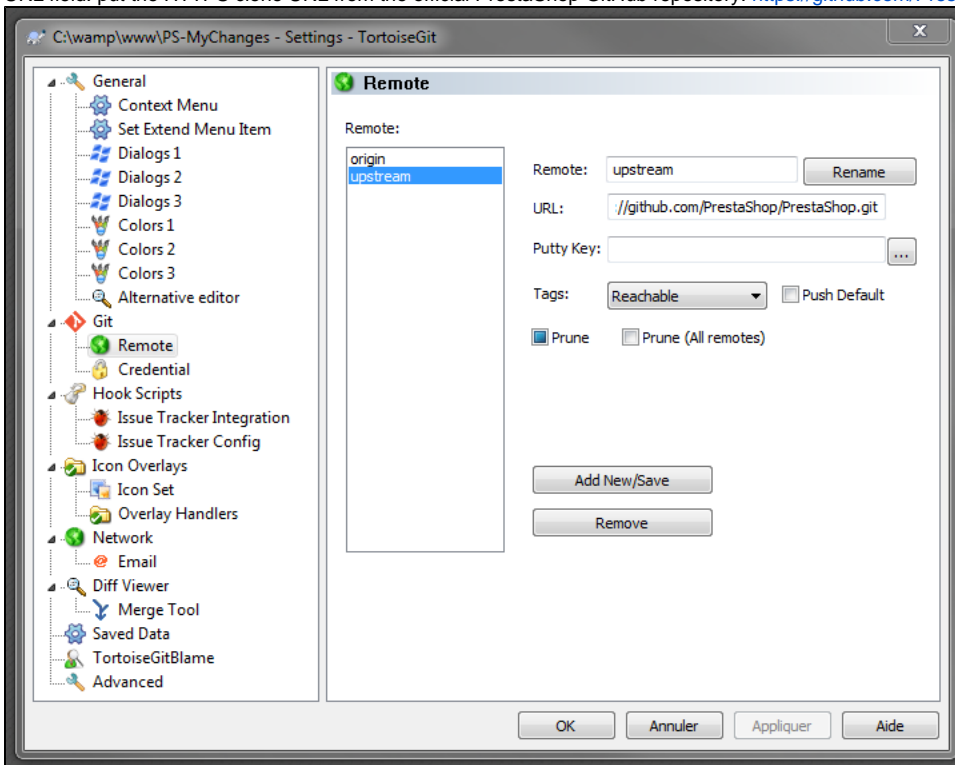


## Keeping your fork and clone up to date

If you plan on contributing to PrestaShop for anything more than a very quick fix, you should make your fork track the official PrestaShop repository. This is what you should have done in the optional 3.b. step, "Add the 'upstream' remote" – and you can still do it now!

Here is how to set up the local upstream remote if you haven't already:

1. Right-click on the PrestaShop folder in the PS-MyChanges folder (it should have a green mark if no change has been made yet), and at the bottom of the TortoiseGit submenu, choose "Settings". This opens the Settings panel.
2. Open the Git > Remote section of the Settings panel.
3. Remote field: Type "upstream".
4. URL field: put the HTTPS clone URL from the official PrestaShop GitHub repository: <https://github.com/PrestaShop/PrestaShop.git> .



5. Click the "Add new/Save" button. TortoiseGit asks you if you want to disable tag fetching for this remote: choose "No", since this remote IS an official remote.
6. TortoiseGit asks you if you want to fetch remote branches from the newly added remote: choose "Yes".
7. TortoiseGit displays the Fetch window for PS-MyChanges: click "OK". A window will open showing that Git is fetching data. Close it once it is done, and close the still-open Settings window (which should by now list "upstream" below "origin" in the list of remote URLs).

### Now you can update the local clone:

1. Right-click on the PS-MyChanges folder, and choose "Git sync..."
2. In the Git Synchronization window that appears, choose "upstream" as the remote URL, and click on "Pull".
3. TortoiseGit displays status information, downloads the latest data from your upstream remote URL (which should be <https://github.com/PrestaShop/PrestaShop.git>), and finally displays the latest changes from upstream, which have now been merged to your local repository.

Your local clone is now up to date, but you might want your fork on GitHub to also be on par with the upstream repository. You cannot do this from one GitHub repo to another, but TortoiseGit can do the trick for you, right after you have updated your local clone:

1. Open the Git Synchronization window if it is not yet open: right-click on the PS-MyChanges folder, and choose "Git sync..."
2. Choose "origin" as the remote URL, and click on "Push".
3. TortoiseGit updates the window with the current status, and might also open an OpenSSH window asking you for your GitHub username, then your GitHub password. Give them both and validate: the status window should push your changes to your GitHub repository.

You're done!

### Making further pull requests for PrestaShop

You can continue editing your local copy of PrestaShop and submit your changes to the PrestaShop developers with the process we explained earlier. Here it is again, in a complete form:

1. On your local machine:
  - a. Always pull the latest changes from the upstream repository. This will prevent bad surprises, where you cannot push your changes until you have merged the ones that were made by other developers!
  - b. Make your changes:
    - i. Edit all the necessary files. As soon as you save and edited file, Windows will change its icon from green to red to indicate that it has been changed since the last update.

- c. Submit your changes to your GitHub repository:
  - i. Once you are done editing the local files, reach the root of your local repository (the folder above PS-MyChanges).
  - ii. Right-click on the PS-MyChanges and choose "Git Commit -> develop", with "develop" being your current branch. This will open an interface window. The files you have changed should appear in the "Changes made" section.
  - iii. Write a commit message for your changes. Be sure to follow [the commit message norm](#).
  - iv. Click the "OK" button: your changes are now committed to your local Git repository. Click "Close".
  - v. Right-click on the PS-MyChanges and choose "Git Sync...": a window opens, asking you to which branch the local commit should be pushed – usually it should be "develop". Leave "origin" as the remote URL, and click "Push".
  - vi. TortoiseGit updates the window with the current status, and might also open an OpenSSH window asking you for your GitHub username, then your GitHub password. Give them both and validate: the status window should push your changes to your GitHub repository (see above).
  - vii. Success! You can close the TortoiseGit windows: the rest will happen on GitHub.
2. Submit your changes to the PrestaShop repository:
  - a. Go to your own PrestaShop repository on GitHub.
  - b. Either click on the "Pull Request" or the "Compare" button. Both will open the same screen, comparing your latest commit to the original code.
  - c. The top of the screen indicates what is being compared. It should be PrestaShop's "develop" branch with your own "develop" branch. You should not have anything to change here, just make sure that the base fork is the PrestaShop repository, and that your own repository is the head fork.
  - d. Click on the "Create a pull request" button. A new screen should appear, in which you can review your changes and edit your commit message (it should already use the commit message you used from your local clone).
  - e. Click on the "Send pull request" button: GitHub open a screen where you can start a conversation about your pull request, right in PrestaShop's GitHub repository. Congratulations, your changes are now ready to add to the PrestaShop codebase!
3. Wait for one of the core developers either to include your change in the codebase, or to comment on possible improvements you should make to your code.

Welcome to the fabulous world of Git collaboration!

## Making a pull request for one of PrestaShop's modules

PrestaShop's default modules are stored in separate repositories from the main repository: each has its own repository on PrestaShop's GitHub account. For instance:

- 1-Click Upgrade: <https://github.com/PrestaShop/autoupgrade>
- HomesModule: <https://github.com/PrestaShop/homeslider>
- Theme Configurator: <https://github.com/PrestaShop/themeconfigurator>
- etc.

You can see the various repositories here: <https://github.com/PrestaShop>

They are separated on the main repository, but since you made a recursive clone of PrestaShop, they were all downloaded by Git as if they were part of PrestaShop. You **MUST** do another step in order to have a proper setup:

1. Right-click on your clone folder (in this example, PS-MyChanges) and choose the "Git bash here" option. If you do not have it, install Git for Windows (<http://msysgit.github.io/>) with "Git bash here" option enabled.
2. In the bash window that opens, type this command: `git submodule update --init` (without the quotes).

This will pull the modules and detach them. They won't have any branch: this is meant to avoid breaking your project when a third-party submodule is updated.

Having modules separated from the main PrestaShop repository means that you cannot push edited module files into the main repository; you have to follow a process similar to the one that you used for PrestaShop:

1. (optional) Fork the module's repository to your own GitHub account. For instance, fork <https://github.com/PrestaShop/gridhtml> to your own <https://github.com/your-username/gridhtml>.
2. Add your fork as an 'origin' remote on the module's local folder:
  - a. Go to your local modules directory (for instance, `/PS-MyChange/modules`).
  - b. Open TortoiseGit's "Settings" window for the module's folder (for instance, `/gridhtml's folder`).
  - c. Go to the Git > Remote section. You should already have one registered remote URLs: "origin", pointing to the original repository for the module (for instance, <https://github.com/PrestaShop/gridhtml.git>).
  - d. Create an "upstream" remote, while keeping the URL as-is. Validate, and retrieve the branches.
  - e. Create a new "origin" URL for your fork's URL. For instance, with gridhtml, that would be <https://github.com/your-username/gridhtml.git>.
  - f. Validate.

You are now ready to edit your files, but please understand this: you should make your changes (or at least push them) to the "dev" branch of the module, **NOT** the "master" branch. The PrestaShop

developers will not accept changes sent to the "master" branch! Here is how to switch branch with TortoiseGit:

1. Right-click on the module's folder, and in the TortoiseGit submenu, choose "Switch/Checkout...".
2. In the window that opens, choose the "remotes/origin/dev" branch and click "OK".

There, you can make your changes!

Now, when pushing your changes:

1. Right-click on the module's folder, and choose "Git sync...".
2. Make sure you are pushing the local dev branch to the remote dev branch, and that the remote URL is "origin".
3. Click "Push".

Your changes are now on your GitHub account, in the repository for this module (for instance, <https://github.com/your-username/gridhtml>). All you have to do now is create a Pull Request for these changes, just as you would do for PrestaShop.

Thank you for your contribution!

## Understanding Git

The GitHub website has more information [on how to fork a repo](#), [how to use pull requests](#) and [many more articles on GitHub collaboration](#), [advanced Git usage](#) and [managing Git remotes](#), but there are many ways to learn more about Git and GitHub:

- The Git book: <http://git-scm.com/book>
- git ready: <http://gitready.com/>
- Git Immersion: <http://gitimmersion.com/>
- Think Like (a) Git: <http://think-like-a-git.net/>
- Git Cheat Sheet: <http://byte.kde.org/~zrusin/git/git-cheat-sheet-medium.png>
- Try GitHub: <http://try.github.io/>
- GitHub help: <https://help.github.com/>
- Rich Jones: [How to GitHub: Fork, Branch, Track, Squash and Pull Request](#)

## Making changes directly in GitHub

**This is not the recommended way to contribute code**, and should only be seen as alternative if you don't feel comfortable forking the project or using Git altogether. One of the issues of this method is that it might bring hundreds of untested pull requests to the PrestaShop developers, who are already trying hard to focus on their own assigned projects...

GitHub makes it possible to edit files directly online. Here is how:

1. Browse to the target file on [PrestaShop's repository](#). Make sure you are on the "1.6" branch!
2. Click the "Edit" button: GitHub will open a web-based editor, indicating that they are taking care of forking the project for you.
3. Make your changes in the editor. Be sure to follow the [PrestaShop coding standard](#)!
4. Once done, click on "Propose file change".

GitHub will then automatically create a new fork with a "develop" branch on your own GitHub account, then suggest to create a pull request to PrestaShop: just write a commit message (follow the [PrestaShop commit message norm](#)!), and validate. Once the pull request is submitted, you just have to wait for a core developer to act on it.

You can also easily create a new file directly on GitHub:

1. From your fork of PrestaShop, click the "+" button next to the project's name (usually, next to "PrestaShop").
2. Once the online editor is open, type your code in the screen. Try not to forget to add the notice of license and the disclaimer in a comment section at the top of the file, just like every other files in PrestaShop do.
3. Save your file. Do not forget to give it a name, along with a commit summary and, if needed, an extended description of what your file brings to PrestaShop. This is essential as the PrestaShop developer will use it to get the context of your file, and therefore to decide whether it should be added to PrestaShop or not.



Make sure that you are always making changes to the latest version of the original repository. This means that the "Branch" button above the list of file should read "branch: develop", and **not** "branch: master":

- The "develop" branch is the one where all current works for PrestaShop 1.7 are pushed to.
  - The "1.6.1.x" is used solely for fix on the "1.6" version of PrestaShop
- The "master" branch is a snapshot of the development branch which was used to build the previous release of PrestaShop.
- The "release" branch (if available) is the forthcoming release.
- Others branches are for specific current projects, and should not be touched.