> *This wiki is automatically published from ohmyzsh/wiki. To edit this page, go to ohmyzsh/wiki, make your changes and submit a Pull Request.*

Oh My Zsh is a community-driven project and contributions are welcome, but it all works best with a proper setup. Here follow the best practices for setting up a fork and getting started for contributing.

Prerequisites

You will of course need Git and a UNIX terminal. You should be familiar with the basics of contributing on GitHub.

You will have to make a choice, whether you simply want to properly send PRs, or if you also wish to maintain your own fork (with your own changes) as a personal alternative version of OMZ. Think about it, it will come around later on.

The following section explains how to set up your local Oh My Zsh install with the proper remote repositories.

Setting up

From this point on, `<name>` stands for your GitHub username.

- **1.** Fork the original repository (button at the top of the repo's page).

This supposes that you have Oh My Zsh installed already in `$ZSH` (default: `~/.oh-my-zsh`) with the origin remote pointing to the original repository.

- **2.** Now make the choice: either we keep it easy and simple, need-to-know basis, or you want a fully maintained fork with your own changes. Here follows a binary dichotomy:
  - **Simple contribution:** auto-upgrade works, and you can send clean PRs; but your fork is not maintained and you are not using your own master branch
  - **Maintained fork:** you can send clean PRs, maintain your own fork with changes, use and share your master branch; but auto-upgrade will not work

Simple contribution setup

All the following uses `<name>` for the remote name, but you can use something else.

- **3.** Go to your local install directory and add your own repository as a remote

```
cd $ZSH
git remote add <name> git@github.com:<name>/oh-my-zsh.git
```

Upgrading: as long as your local master branch cleanly follows origin/master (which still points to the original repository), auto-upgrade should work fine.

More setups: on a new machine, simply install OMZ as usual, do step 3, and of course copy your startup files; you will be able to get upgrades, and to work on your pending PRs.

Sending PRs

Maintained fork setup

- **3.** Go to your local install directory and rename the origin remote to "upstream"

```
cd $ZSH
git remote rename origin upstream
```

- **4.** Then set your own repository as the origin remote

```
git remote add origin git@github.com:<name>/oh-my-zsh.git
```

Upgrading:

- **Upstream:** when you want to get the latest upgrades from the original repository (*aka* upstream), simply `git pull upstream master`; you may have to solve conflicts with your own changes of course; when you are satisfied with the update you can `git push origin master`.
- **Origin:** when you want to get the latest upgrades from your own fork (*aka* origin), simply `git pull --rebase origin master`; you may have to solve conflicts with your local changes of course; when you are satisfied with the update you can `git push --force origin master`.

More setups: on a new machine, simply `git clone git@github.com:<name>/oh-my-zsh.git ~/.oh-my-zsh` and `git remote add upstream git@github.com:ohmyzsh/ohmyzsh.git`, and of course copy your startup files; you will be able to get upgrades, and to work on your pending PRs.

Sending PRs

Pull Requests

These are technical instructions, please read the contributing guidelines (project) before sending PRs.

The following uses `my-new-pr` *for the branch name, but you can use something else.*

Simple contribution PRs

- **1.** Any new PR must start from a clean upstream tree

```
git checkout origin/master
git checkout -b my-new-pr
```

You are now on your dedicated PR branch. Time to commit some changes!

- **2.** Send your commits

```
git push <name> my-new-pr
```

You can now go to GitHub and create the PR.

Updating: in case there are conflicting changes upstream after you created the branch, you will have to `git pull --rebase origin master` on `my-new-pr` branch and resolve conflicts, and then `git push --force <name> my-new-pr`, in order to keep your PR up to date.

More setups: on another machine, simply `git checkout <name>/my-new-pr` and `git checkout -b my-new-pr` to continue working on a pending PR.

Maintained fork PRs

- **1.** Any new PR must start from a clean upstream tree

```
git checkout upstream/master
git checkout -b my-new-pr
```

You are now on your dedicated PR branch. Time to commit some changes!

- **2.** Send your commits

```
git push origin my-new-pr
```

You can now go to GitHub and create the PR.

Updating: in case there are conflicting changes upstream after you created the branch, you will have to `git pull --rebase upstream master` on `my-new-pr` branch and resolve conflicts, and then `git push --force origin my-new-pr`, in order to keep your PR up to date.

More setups: on another machine, simply `git checkout origin/my-new-pr` and `git checkout -b my-new-pr` to continue working on a pending PR.