# 使用 **yadm** 管理点文件

孔俊

2023-04-07

# Contents

> *When you live in a command line, configurations are a deeply personal thing. They are often crafted over years of experience, battles lost, lessons learned, advice followed, and ingenuity rewarded. When you are away from your own configurations, you are an orphaned refugee in unfamiliar and hostile surroundings. You feel clumsy and out of sorts. You are filled with a sense of longing to be back in a place you know. A place you built. A place where all the short-cuts have been worn bare by your own travels. A place you proudly call⋯$HOME.*
>
> -- yadm website

## 1 点文件管理的痛点

管理点文件主要有两个目的：

- 方便拷贝配置到新机器
- 方便地查看对配置的更改

对于软件项目，Git 结合 Github 可以很好的满足这两点需求。但点文件不像软件代码集中在一个目录中，因此不能直接`git init`创建仓库并管理。

为了把 Git 引入点文件管理，通常会创建一个裸仓库（bare repository，即没有工作区的 Git 仓库），然后在这个 repo 中添加点文件。使用 Git 时，再通过`git --work-dir=${HOME} --git-dir=<bare repo>`临时把家目录设置为工作区。

Git bare repo 的技巧让点文件管理进入了版本控制时代：

1. 创建裸仓库：`git init --bare ~/.dotfiles`；
2. 设置 Git wrapper：`alias dotfiles="git --git-dir=$HOME/.dotfiles --work-tree=$HOME"`；
3. 将`git`替换为`dotfiles`，一切照旧。

`dotfiles`虽然让点文件管理越过了原始时代，但还有以下问题：

1. 编辑器支持不足。
2. 新机器上克隆配置会出现冲突，不得不编写脚本备份旧配置并强制覆盖。

## 2 yadm！

yadm(Yet Another Dotfiles Manager)是究极进化版的`dotfiles`，是完美的 Git wrapper！

用 yadm，不需要手动初始化 bare repo，不需要为新机器上出现的 Git conflicts 头疼：

1. `yadm init`创建空仓库；

2. 整个家目录都在仓库中，`yadm add <file>`添加点文件；
3. `yadm commit`提交后，用`yadm push`推送到 Github；
4. 切换到新机器，`yadm clone <remote repo>`点文件就会干干净净地出现在系统中！

`yadm clone`的神迹实际上是利用了`git stash`，`yadm clone`将整个家目录视作 Git repo，并`git stash`旧配置，优雅地解决了第 2 点问题。

## 3 集成到 Vim/Neovim

gitsigns.nvim支持 yadm：

```lua
require('gitsigns').setup {
  yadm = {
    enable = enable
  },
}
```

使用 yadm-git.vim 为vim-fugitive 和 vim-gitgutter 添加 yadm 支持。

## 4 解决 yadm 兼容性问题

yadm 新旧版本存在一定兼容性问题，主要是：

1. 旧版本不识别名为 main 的主分支。
2. 新旧版本无法互相识别 Git 仓库。

旧版 Git 的默认主分支名为 master，而新版本的默认主分支名为 main，因此旧版本 yadm 无法识别 main 分支。克隆仓库是使用`-b main`参数来指定主分支即可。

新版 yadm 遵守 XDG Base Directory Specification 规范，Git 仓库被放置在 ~/.local/share/yadm.repo.git 中，而旧版放在不同位置。因此新旧版本无法互相识别 Git 仓库，在 bashrc/zshrc 中添加命令别名`alias yadm="yadm --yadm-repo ~/.local/share/yadm/repo.git"`指明路径即可。

## 5 yadm bootstrap

克隆配置后，往往还要初始化系统，为此 yadm 还提供了`yadm bootstrap`命令。该命令执行程序 ~/.config/yadm/bootstrap，这个文件可以是任何可执行的代码。可以专门写一个 bootstrap 脚本，负责安装 zsh 插件、必要的开发工具等系统初始化工作。

yadm 支持钩子，可以为每个命令提供一个钩子在运行前或运行后运行。钩子位于 ~/.config/yadm/hooks 目录，名为 post_ 或 pre_。可以设置一个 post_clone 钩子，在克隆仓库后执行`yadm bootstrap`。

好的 bootstrap 脚本要实现幂等性，并尽量鲁棒，以避免出错导致系统处于不干净的状态。为此，要注意以下几点：

- 将整个 bootstrap 过程的输出保留在日志中
- 任何一步失败都直接退出
- 避免重复操作
- 优雅退出：退出时要清理中间产物
- 明确操作系统版本

更方便的做法是编写一个完成所有配置操作的 bootstrap.sh，在新机器上直接执行该脚本，免去繁琐的安装配置过程。我的 bootstrap.sh 支持 Ubuntu/Debian，system-bootstrap.sh 根据参数调用 bootstrap.sh 初始化本地机器或远程机器（通过 ssh）。

## 6 其他功能

- 模版（Template）：配置文件写成模版，克隆到本地后生成需要的配置文件。
- 备用文件（Alternate file）：根据条件，将配置文件（符号链接）指向适当版本。
- 加密（Encryption）：支持文件加密解密，以便安全地托管 SSH key 等保密文件。

这些功能相对鸡肋，感兴趣地话可以阅读文档。

模版和备用文件用于实现跨平台的配置文件，但更好的方式是直接编写跨平台的配置。

加密用于安全地托管保密文件，但最安全的方式是不托管保密文件。