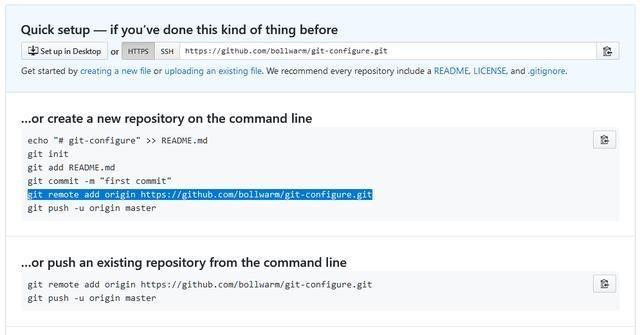
Git是一个非常酷的工具，截止目前已经远远超越了其作为版本管理（VCS）的初衷，成了集版本管理、项目管理、开发流程、CI/CD,DevOps,协作社交为一体的综合性工具了。Git的开始也很简单，只需安装一个客户端，注册一个公共服务器账号（如Github,Gitee等）、Web下创建一个项目。clone项目到本地，然后就开始了。刚开始只需学一些简单的命令：clone、add，commit，checkout，push，pull，status和diff然后就足矣应付你日常的工作了。随着工作的深入，你可能需要学习分支、tag标签、等操作和概念。你使用git越多，你就会学习能学习到更多的git命令宝藏，比如blame, bisect,rebase、reset，reflog，submodules, hooks等等，这些工具的使用和概念介绍等，虫虫之前的文章中曾经做过介绍，关注虫虫可以看得到。



在git使用和我们的gitlab群中有一个常出现的问题，让很多初学者迷惑，那就是关于git的配置。为此今天虫虫专门撰写本文来介绍下git的基本配置，以帮助大家解疑你遇到一些莫名其妙的问题，答惑为什么会出现这样的问题。而实际中这些的出现仅仅是因为你使用了不一样的配置，而你却不知道。

**git的配置文件**

当你在git服务器Web管理页面创建完第一个项目的时候，接下来的一个页面会提示你做一些基本的git配置，比如github的提示页面如下：



你按照提示这些命令中有一个蓝色标注的git remote add命令。实际提到这些命令配置还比较繁琐，你可以用clone在你的机器上直接创建一个空项目，忽略到上面提示中这许多步骤。

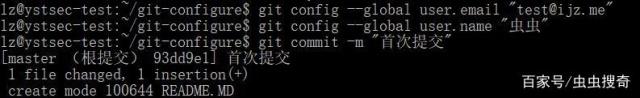


git commit提交历史时候，提示你需要配置global参数，设置你的邮箱地址和名称（注意这个和你git服务器的账号可以不同哦，仅仅用于在git历史中显示）：

git config --global user.email"you@example.com"

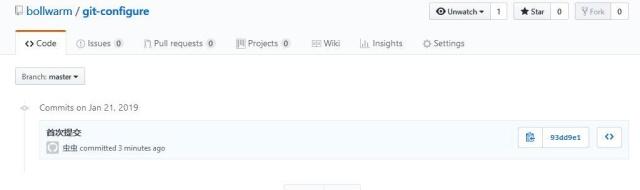
git config --global user.name"你的名字"

运行这两个命令后，你就可以执行commit



然后你push推送到github

你设置的用户名，比如此例中的"虫虫"会出现在commit历史中，看github提交历史：



注意该过程需要你github的用户名和密码认证一下，该用户名也可以缓存在本地的配置文件中（[credential]部分），如果你用户名输入有误，则需要清除这个，不然后续认证会有问题。

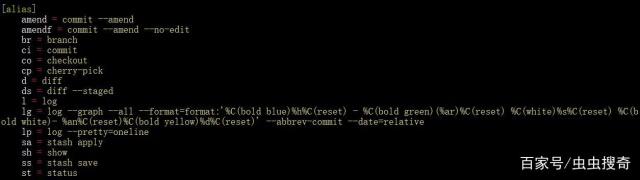
上面提到的git remote add 和git config设置的参数是加上都是配置在git的配置文件中的。其中git remote添加的仓库Url链接信息是保存在项目目录.git/config文件中，这是项目级的配置文件。而git config设置的global信息保存在用户级的配置文件中，该文件的位置为用户目录~/.gitconfig。实际上git还有一个系统级的配置文件，该文件位于/etc/gitconfig。配置文件生效生效优先级是先项目（.git/config），后用户(~/.gitconfig),最后系统（/etc/gitconfig）。由于git的使用都是以用户级的，所以我们日常用的最多都是通过用户级别的配置，本文主要是用户级别配置文件的介绍。

**git配置文件详解**

**别名**

git别名是我们日常进行git配置使用最多的一部分内容。在git使用中，为了便捷，减少输入，git提供了别名机制来将标准的git命令自定义为自己习惯使用的命令。我们可以将git的命令设定别名为为1或2个字母的快捷方式。

例如：



[alias]

amend = commit --amend

amendf = commit --amend --no-edit

br = branch

ct = commit

co = checkout

cp = cherry-pick

df = diff

ds = diff --staged

l = log

lg = log --graph --all --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(bold white)— %an%C(reset)%C(bold yellow)%d%C(reset)' --abbrev-commit --date=relative

lp = log --pretty=oneline

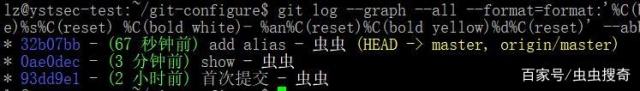
sa = stash apply

sh = show

ss = stash save

st = status

此处我们按字母顺序排列别名。注意到这个log命令lg，我们美化了log的显示，是的lgo显示更加好看美，简洁的图表，说明了仓库库随着时间的推移如何演变。



**配色**

给你的工作终端设置一个好看的颜色也是每一个码农一直孜孜以求的目标，那就看本部分：

[color]

ui = auto

[color "branch"]

current = yellow reverse

local = yellow

remote = green

[color "status"]

added = yellow

changed = green

untracked = cyan

[color "diff"]

meta = yellow

frag = magenta bold

commit = yellow bold

old = red bold

new = green bold

whitespace = red reverse

[color "diff-highlight"]

oldNormal = red bold

oldHighlight = red bold 52

newNormal = green bold

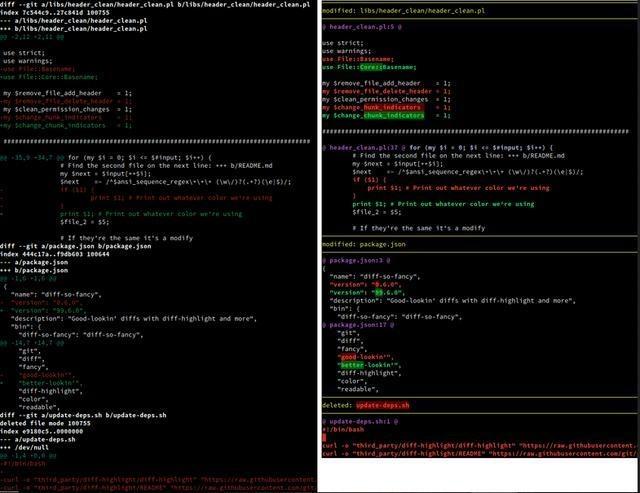
newHighlight = green bold 22



ui = auto是UI的默认设置。它在输出直接到终端时为输出着色，但在输出重定向到管道或文件时会省略颜色控制代码，以免导致问题。分支和状态部分正在以下列方式更改git branch和git status命令的输出颜色。



diff和diff-highlight设置将在执行diff命令时候系统展示的颜色。对于diff，也可以使用一些第三方的插件，比如diff-so-fancy工具，显示更加好看，下面是diff-so-fancy工具官方库主页的介绍对比截图：



**核心**

[core]

editor = vim

excludesfile = ~/.gitignore

pager = diff-so-fancy | less --tabs=4 -RFX

autocrlf = input

核心设置部分包含与git相关的各种不同设置。我们设置到的部分有：

editor = vim设置要用于编辑提交消息的编辑器（如果未设置此值，git将首先尝试从环境变量VISUAL或EDITOR读取你当前的编辑器，如果获取不到，最终会使用vi）。

excludesfile =~/.gitignore允许指定全局性质的.gitignore文件。每个git存储库都可以设置特定的项目级别的.gitignore文件，该文件指定要从版本控制中排除的文件。但很多时候，每个git存储库中的一些文件都是相同的（例如，macOS上的.DS\_Store，或者当你是Python开发人员时是\*.pyc），为了避免重复设置，可以设置全局性质的.gitignore，该设置就会对该用户下所有的项目都生效。

pager = diff-so-fancy | less --tabs = 4 -RFX指定要用于显示git log，git diff和git show命令输出的工具。默认情况下，git使用的是less。该配置中我们指定使用diff-so-fancy。

autocrlf = input。 由于Windows使用的是与Unix和MacOS不同的行结尾，如果来自不同操作系统的人员提交到同一个存储库，则可能会造成一些混乱。关于换行设置三种操作系统（windows，linux和macOS）的是不一样的：

MacOS/Linux设置：autocrlf = input

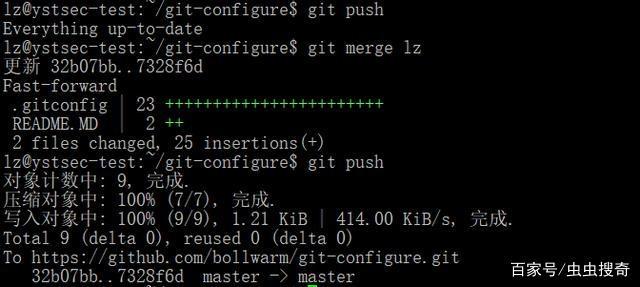
Windows上的autocrlf = true）。

**凭据**

[credential]

helper = cache --timeout=28800

凭据部分用来指定希望你需要默认保存的账号和密码。默认情况下，git根本不会包存储凭据，所以，每次连接时（http（s）性质的仓库）都会提示输入用户名和密用,这会很麻烦，你可以在此处设置保存户名密码，这样就可以不用每次都输入了（当然最好方式，是用ssh证书方法，即安全又方便）。凭据一般可以通过两种方式保存，保存在带有store选项的文件中（它将使用你的凭据创建纯文本文件），或使用cache选项将它们存储在内存中。当然根据你系统不同，还可以使用第三方的方式，比如MacOS下的osxkeychain，Windows下的Git Credential Manager）。此处我们使用的是，通过内存cache的方式，默认是15分钟，此处我们设置为8小时。



设置后，在输入用户名密码一次后，再push就不需要认证了。

**推送**

[push]

default = current

git push命令中包含分支的名称，如果你没有添加，可能导致意外的行为（例如我正在开发dev分支，但不小心，push到了master分支）。为了防止这种错误，我给push设置了default = current选项。现在，如果忘记包含分支的名称，git将尝试推送到具有相同名称的分支。如果它在远程库中找不到具有相同名称的分支，会新创建一个。

本文中，我们给出了git常见设置，示例介绍了从创建项目到对你git环境的配置，.gitconfig配置文件和设置的方法。更多的信息可以参考git-config官方文档页面，他有更详细的选贤说明。



当然获取配置文件最好的方法莫过搜索引擎了，我给你关键字"gitconfig dotfiles"，其他的就看需要什么样的了。本文所有配置示例文件，见我们上面截图中的github的git-configure项目，大家可以参考学习。