GitHub CLI 快速参考

仓库、issue、PR、Actions、发布、API

初始化

安装

`brew install gh`	macOS via Homebrew
`sudo apt install gh`	Debian / Ubuntu
`winget install GitHub.cli`	Windows via winget
`conda install gh`	via conda-forge

认证

```
gh auth login                # interactive login
gh auth login --with-token < token.txt
gh auth status               # check auth state
gh auth refresh -s repo,gist   # add scopes
```

配置

```
gh config set editor vim
gh config set pager less
gh config set git_protocol ssh
gh config list
```

仓库

仓库命令

```
gh repo create my-app --public --clone
gh repo clone owner/repo
gh repo fork owner/repo --clone
gh repo view owner/repo --web
```

仓库选项

`--public | --private`	设置仓库可见性
`--template owner/repo`	从模板仓库创建
`--clone`	创建后立即 clone
`--add-readme`	初始化时添加 README
`gh repo list owner`	列出 owner 的仓库
`gh repo delete owner/repo`	删除仓库（需确认）
`gh repo rename new-name`	重命名当前仓库
`gh repo archive owner/repo`	归档仓库

Issue

管理 Issue

```
gh issue create --title "Bug" --body "Details here"
gh issue list --state open --label bug
gh issue view 42
gh issue close 42 --reason completed
```

Issue 选项

`--assignee @me`	指派给自己
`--label bug,urgent`	添加标签
`--milestone v2.0`	设置里程碑
`--project "Board"`	添加到项目
`gh issue edit 42`	交互式编辑 issue
`gh issue reopen 42`	重新打开已关闭的 issue
`gh issue comment 42 -b "msg"`	给 issue 添加评论
`gh issue pin 42`	将 issue 置顶

Pull Request

创建与管理 PR

```
gh pr create --title "feat: add auth" --body "..."
gh pr create --fill          # title/body from commits
gh pr list --state open
gh pr view 123 --web
```

审查与合并

```
gh pr checkout 123           # check out PR branch
gh pr diff 123               # view PR diff
gh pr review 123 --approve
gh pr merge 123 --squash --delete-branch
```

PR 选项

`--draft`	创建为草稿 PR
`--reviewer user1,user2`	请求审阅者
`--base main`	设置基础分支
`--merge | --squash | --rebase`	合并策略
`--auto`	检查通过后自动合并
`--delete-branch`	合并后删除分支
`gh pr ready 123`	将草稿 PR 标记为就绪

Actions

工作流命令

```
gh run list                      # recent runs
gh run view 12345                # run details
gh run view 12345 --log-failed   # failed step logs
gh run watch 12345               # live status
```

触发与管理

```
gh workflow run deploy.yml --ref main
gh workflow list
gh workflow view deploy.yml
gh run rerun 12345 --failed   # rerun failed jobs
```

Actions 选项

`-f key=value`	向 workflow_dispatch 传入参数
`--json`	以 JSON 格式输出
`-b branch`	按分支过滤
`gh run download 12345`	下载运行产物
`gh cache list`	列出 Actions 缓存
`gh cache delete KEY`	删除缓存条目

发布

管理发布

```
gh release create v1.0.0 --generate-notes
gh release create v1.0.0 ./dist/*.tar.gz
gh release list
gh release view v1.0.0
```

发布选项

`--title "Release v1.0"`	设置发布标题
`--notes "Changelog here"`	内联设置发布说明
`--notes-file CHANGELOG.md`	从文件读取说明
`--generate-notes`	从 commit 自动生成
`--draft`	创建为草稿
`--prerelease`	标记为预发布
`--latest`	标记为最新发布
`gh release download v1.0.0`	下载发布资产
`gh release delete v1.0.0`	删除发布
`gh release edit v1.0.0`	编辑发布元数据

Gist

Gist 命令

```
gh gist create file.py -d "My snippet"
gh gist create file1.js file2.js   # multi-file gist
gh gist list
gh gist view <id>
```

Gist 选项

`-d "description"`	设置 gist 描述
`--public`	创建公开 gist（默认：私密）
`--web`	在浏览器中打开
`gh gist edit <id>`	编辑 gist 文件
`gh gist clone <id>`	本地克隆 gist
`gh gist delete <id>`	删除 gist

API

发起 API 请求

```
gh api repos/owner/repo
gh api repos/owner/repo/issues --method POST \
  -f title="Bug" -f body="Details"
gh api graphql -f query='{ viewer { login } }'
```

格式化输出

```
gh api repos/owner/repo --jq '.stargazers_count'
gh api repos/owner/repo --template '{{.full_name}}'
gh pr list --json number,title --jq '.[].title'
```

API 选项

`--method GET|POST|PUT|DELETE`	HTTP 方法
`-f key=value`	设置字符串字段
`-F key=@file`	从文件读取字段
`--jq 'expression'`	用 jq 语法过滤 JSON
`--template 'tmpl'`	用 Go 模板格式化
`--paginate`	获取所有分页
`-H 'Accept: ...'`	设置自定义头

别名

管理别名

```
gh alias set co 'pr checkout'
gh alias set bugs 'issue list --label bug'
gh alias set last 'run list -L 1'
gh alias list
```

高级别名

```
# Alias with shell command
gh alias set --shell pv 'gh pr view --json url --jq .url | pbcopy'

# Delete alias
gh alias delete co
```

别名技巧

`gh alias set name 'cmd'`	创建简单别名
`--shell`	通过 shell 执行（支持管道）
`gh alias list`	显示所有已定义别名
`gh alias delete name`	删除别名

常用模式

日常工作流

```
gh issue list --assignee @me       # my open issues
gh pr status                       # PRs needing attention
gh pr checks 123                   # CI status for PR
gh run list -b main -L 5           # recent CI runs
```

搜索

```
gh search repos --language rust --stars ">1000"
gh search issues --repo owner/repo "memory leak"
gh search prs --state open --review required
```

GitHub CLI 快速参考

常用 Flag

`--json field1,field2`	以 JSON 输出指定字段
`--jq 'expr'`	过滤 JSON 输出
`-L N`	限制结果数量
`--web`	在浏览器中打开结果
`-R owner/repo`	指定目标仓库
`GH_TOKEN=xxx`	通过环境变量认证