GITHUB CLI クイックリファレンス

リポジトリ、イシュー、PR、Actions、リリース、API

セットアップ

インストール

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

認証

```
gh auth login                  # 対話形式でログイン
gh auth login --with-token < token.txt
gh auth status                 # 認証状態を確認
gh auth refresh -s repo,gist   # スコープを追加
```

設定

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

リポジトリ

リポジトリコマンド

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

リポジトリオプション

`--public | --private`	リポジトリの公開設定
`--template owner/repo`	テンプレートリポジトリから作成
`--clone`	作成後にクローン
`--add-readme`	README で初期化
`gh repo list owner`	オーナーのリポジトリを一覧表示
`gh repo delete owner/repo`	リポジトリを削除（確認あり）
`gh repo rename new-name`	現在のリポジトリの名前を変更
`gh repo archive owner/repo`	リポジトリをアーカイブ

イシュー

イシューの管理

```
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
```

イシューオプション

`--assignee @me`	自分に割り当て
`--label bug,urgent`	ラベルを追加
`--milestone v2.0`	マイルストーンを設定
`--project "Board"`	プロジェクトに追加
`gh issue edit 42`	イシューを対話形式で編集
`gh issue reopen 42`	クローズされたイシューを再開
`gh issue comment 42 -b "msg"`	イシューにコメントを追加
`gh issue pin 42`	イシューをリポジトリにピン留め

プルリクエスト

PR の作成と管理

```
gh pr create --title "feat: add auth" --body "..."
gh pr create --fill          # コミットからタイトル/本文を取得
gh pr list --state open
gh pr view 123 --web
```

レビューとマージ

```
gh pr checkout 123           # PR ブランチをチェックアウト
gh pr diff 123               # PR の差分を表示
gh pr review 123 --approve
gh pr merge 123 --squash --delete-branch
```

PR オプション

`--draft`	ドラフト PR として作成
`--reviewer user1,user2`	レビュアーをリクエスト
`--base main`	ベースブランチを設定
`--merge | --squash | --rebase`	マージ戦略
`--auto`	チェックが通ったら自動マージを有効化
`--delete-branch`	マージ後にブランチを削除
`gh pr ready 123`	ドラフト PR をレビュー可能にする

Actions

ワークフローコマンド

```
gh run list                  # 最近の実行
gh run view 12345            # 実行の詳細
gh run view 12345 --log-failed   # 失敗したステップのログ
gh run watch 12345           # ライブステータス
```

トリガーと管理

```
gh workflow run deploy.yml --ref main
gh workflow list
gh workflow view deploy.yml
gh run rerun 12345 --failed   # 失敗したジョブを再実行
```

Actions オプション

`-f key=value`	workflow_dispatch に入力を渡す
`--json`	JSON として出力
`-b branch`	ブランチでフィルタ
`gh run download 12345`	実行のアーティファクトをダウンロード
`gh cache list`	Actions キャッシュを一覧表示
`gh cache delete KEY`	キャッシュエントリを削除

リリース

リリースの管理

```
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
```

リリースオプション

`--title "Release v1.0"`	リリースタイトルを設定
`--notes "Changelog here"`	リリースノートをインラインで設定
`--notes-file CHANGELOG.md`	ファイルからノートを取得
`--generate-notes`	コミットから自動生成
`--draft`	ドラフトとして作成
`--prerelease`	プレリリースとしてマーク
`--latest`	最新リリースとしてマーク
`gh release download v1.0.0`	リリースアセットをダウンロード
`gh release delete v1.0.0`	リリースを削除
`gh release edit v1.0.0`	リリースのメタデータを編集

Gist

Gist コマンド

```
gh gist create file.py -d "My snippet"
gh gist create file1.js file2.js   # 複数ファイルの gist
gh gist list
gh gist view <id>
```

Gist オプション

`-d "description"`	gist の説明を設定
`--public`	公開 gist を作成（デフォルト: シークレット）
`--web`	ブラウザで gist を開く
`gh gist edit <id>`	gist ファイルを編集
`gh gist clone <id>`	gist をローカルにクローン
`gh gist delete <id>`	gist を削除

API

API 呼び出し

```
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 } }'
```

出力のフォーマット

```
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 オプション

`--method GET|POST|PUT|DELETE`	HTTP メソッド
`-f key=value`	文字列フィールドを設定
`-F key=@file`	ファイルからフィールドを設定
`--jq 'expression'`	jq 構文で JSON をフィルタ
`--template 'tmpl'`	Go テンプレートでフォーマット
`--paginate`	すべてのページを取得
`-H 'Accept: ...'`	カスタムヘッダーを設定

エイリアス

エイリアスの管理

```
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
```

高度なエイリアス

```
# シェルコマンド付きエイリアス
gh alias set --shell pv 'gh pr view --json url --jq .url | pbcopy'

# エイリアスを削除
gh alias delete co
```

エイリアスのヒント

`gh alias set name 'cmd'`	シンプルなエイリアスを作成
`--shell`	シェル経由で実行（パイプ対応）
`gh alias list`	定義済みエイリアスを表示
`gh alias delete name`	エイリアスを削除

よくあるパターン

日常のワークフロー

```
gh issue list --assignee @me        # 自分のオープンイシュー
gh pr status                        # 注意が必要な PR
gh pr checks 123                    # PR の CI ステータス
gh run list -b main -L 5            # 最近の CI 実行
```

検索

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

便利なフラグ

`--json field1,field2`	特定フィールドを JSON で出力
`--jq 'expr'`	JSON 出力をフィルタ
`-L N`	結果を N 件に制限
`--web`	ブラウザで結果を開く
`-R owner/repo`	特定のリポジトリを対象にする
`GH_TOKEN=xxx`	環境変数で認証