

# Explore GitHub — Ultra-Practical Cheat Sheet

Repos, PRs, Actions, Releases, gh CLI, and SemVer you can use today

## 1) Platform overview

**Code** Branches, tags, files, clone URLs, Codespaces.

**Issues** Tasks with labels/milestones; link to PRs and commits.

**Pull Requests**

Code review, checks, merge strategies.

**Actions**

CI/CD: build, test, package, deploy, automate.

**Projects**

Boards/tables/timelines across repos.

**Wiki/Pages**

Documentation and static sites.

**Security**

GHAS: code scanning, Dependabot, secrets, advisories.

**Insights**

Contributors, traffic, forks, clones.

**Settings**

Permissions, branch protection, secrets/variables, webhooks, apps.

**Pro tips.** Star & Watch important repos; use template repos; initialize with README, language .gitignore, and a license.

## 2) Create a repository quickly

1. **New repository** → name, description, visibility.
2. Add README, choose language .gitignore, select a license.
3. **Create repository**, then use **Code** button for HTTPS/SSH/CLI.

### CLI equivalents (GitHub CLI)

---

```
1 # Install: https://cli.github.com
2 gh --version
3 gh auth login
4 gh auth status
5
6 # Create from scratch (public) with README, license, .gitignore
7 gh repo create my-repo \
8   --public --add-readme \
9   --license apache-2.0 \
10  --gitignore Python
11
12 # Clone (HTTPS or SSH depending on your auth)
13 gh repo clone OWNER/REPO
```

---

## 3) Branching & commits

### Branch naming

- feature/short-description, fix/issue-123, docs/update-contrib.
- Keep branches short-lived; rebase or merge from `main` frequently.

### Conventional Commits (recommended)

---

```
1 <type>[optional scope]: <description>
2
3 feat: add search box to header
4 fix(auth): handle expired refresh tokens
5 docs(readme): add quickstart
6 refactor(api): collapse duplicate handlers
7 test(web): add unit tests for navbar
8 chore(deps): bump axios from 1.6.7 to 1.7.0
```

---

## 4) Pull Request workflow (end-to-end)

1. Open an issue or link an existing one.
2. Create a topic branch, commit in small chunks.
3. Open PR: crisp title & body; link issues (Fixes #123).
4. Request review; address comments; keep PR focused.
5. Checks green: unit/integration tests, linters, scanners.
6. Merge: Squash (clean history), Merge commit (context), or Rebase (linear).
7. Delete branch if done; confirm deployment/observability.

## Useful PR CLI

---

```
1 # Create and view
2 gh pr create --base main --title "feat: search box" --body "Adds quick search."
3 gh pr view --web
4
5 # Status, checkout, and merge
6 gh pr status
7 gh pr checkout 123
8 gh pr merge 123 --squash --delete-branch
```

---

## 5) Actions: from zero to useful

### Minimal CI for Node (drop-in)

---

```
1 # .github/workflows/ci.yml
2 name: ci
3 on:
4   push:
5     branches: [ main ]
6   pull_request:
7     branches: [ main ]
8
9 jobs:
10   test:
11     runs-on: ubuntu-latest
12     strategy:
13       matrix:
14         node-version: [18, 20]
15     steps:
16       - uses: actions/checkout@v4
17       - uses: actions/setup-node@v4
18         with:
19           node-version: ${{ matrix.node-version }}
20           cache: npm
21       - run: npm ci
22       - run: npm test --if-present
```

---

## Reusable patterns (matrix, cache, artifacts, concurrency)

---

```
1 # .github/workflows/build.yml
2 name: build
3 on: [push, pull_request]
4
5 concurrency:
6   group: ${{ github.workflow }}-${{ github.ref }}
7   cancel-in-progress: true
8
9 jobs:
10   build:
11     runs-on: ubuntu-latest
12     steps:
13       - uses: actions/checkout@v4
14
15     # Language toolchain example (Node)
16     - uses: actions/setup-node@v4
17       with:
18         node-version: 20
19         cache: npm
20
21     # Build
22     - run: npm ci && npm run build
23
24     # Save build output
25     - uses: actions/upload-artifact@v4
26       with:
27         name: web-dist
28         path: dist/
29         retention-days: 7
```

---

## Environment secrets & variables

- **Secrets:** encrypted (e.g., NPM\_TOKEN); use \${{ secrets.NAME }}.
- **Variables:** non-secret config (e.g., APP\_ENV); use \${{ vars.NAME }}.
- Prefer environment-scoped secrets with required reviewers for production.

## Manual deploy gate with environments

---

```
1 # .github/workflows/deploy.yml
2 name: deploy
3 on:
4   workflow_dispatch:
5
6 jobs:
7   deploy-prod:
8     runs-on: ubuntu-latest
9     environment:
10       name: production
11       url: https://example.com
12     steps:
13       - uses: actions/checkout@v4
14       - run: ./scripts/deploy.sh
15       env:
16         API_TOKEN: ${{ secrets.PROD_API_TOKEN }}
```

---

## 6) Releases & tagging

### Tags vs Releases

- **Tag:** pointer to a commit (`git tag v1.2.0`).
- **Release:** tag *plus* notes, assets, visibility in UI.

### Create release with CLI

---

```
1 # From a prepared CHANGELOG and a tag:
2 git tag v1.2.0
3 git push origin v1.2.0
4
5 # Create a GitHub release (notes from file)
6 gh release create v1.2.0 \
7   --title "v1.2.0" \
8   --notes-file CHANGELOG-1.2.0.md \
9   ./builds/app-linux-x64.tar.gz#linux \
10  ./builds/app-darwin-arm64.tar.gz#mac-arm64
```

---

### Auto-generate release notes

---

```
1 # Let GitHub generate notes based on PRs and commits
2 gh release create v1.3.0 --generate-notes
```

---

## 7) Semantic Versioning (SemVer) in practice

MAJOR breaking changes  
MINOR backwards-compatible features  
PATCH bug fixes / small improvements

### Dependency ranges (common patterns)

- `≥1.4.0` — allow anything newer than/equal to baseline.
- `^1.4.0` — allow MINOR/PATCH updates (no new MAJOR).
- `~1.4.0` — allow PATCH updates within the same MINOR.

## 8) Issues, labels, templates, and Projects

### Issue template (Markdown)

---

```
1 ---
2 name: Bug report
3 about: Create a report to help us improve
4 labels: bug, needs-triage
5 ---
6
7 ### Describe the bug
8 A clear and concise description...
9
10 ### Steps to reproduce
11 1. Go to '...'
12 2. Click on '...'
13
14 ### Expected behavior
15 ...
```

---

### Issue forms (YAML)

---

```
1 # .github/ISSUE_TEMPLATE/bug.yml
2 name: Bug report
3 description: Report a reproducible problem
4 labels: [bug, needs-triage]
5 body:
6   - type: textarea
7     id: description
8     attributes:
9       label: Bug description
10      placeholder: What happened?
11    validations:
12      required: true
13    - type: input
14      id: version
15      attributes:
16        label: Affected version
17        placeholder: e.g., 1.2.3
```

---

### PR template (Markdown)

---

```
1 ## Summary
2 Brief description of changes and why.
3
4 ## Checklist
5 - [ ] Tests added/updated
6 - [ ] Docs updated
7 - [ ] Linked issue: Fixes #123
```

---

## Labels and triage

- Keep a small, meaningful set (e.g., bug, enhancement, docs, deps).
- Use **Projects** for planning and triage dashboards across repos.

## 9) Security essentials (quick wins)

- Enable **Dependabot alerts** and **security updates**.
- Turn on **Secret scanning** (push protection where available).
- Configure **Code scanning** (CodeQL) for primary languages.
- Use **branch protection**: required reviews, required checks, CODEOWNERS.

## CODEOWNERS example

---

```
1 # .github/CODEOWNERS
2 # Order matters; last match wins.
3 *          @org/security-team
4 docs/**    @docs-writers
5 src/web/** @frontend-core
6 src/api/** @backend-core @api-reviewers
```

---

## 10) Advanced `gh` CLI you'll actually use

### Auth, repos, issues, PRs

---

```
1 # Auth
2 gh auth login
3 gh auth status
4
5 # Repos
6 gh repo list my-org --limit 100 --visibility internal
7 gh repo view my-org/my-repo --web
8
9 # Issues
10 gh issue create --title "feat: search facet" \
11   --body "Adds facet by category" --label enhancement
12 gh issue list --label bug --state open
13
14 # PRs
15 gh pr list --search "label:needs-review" --state open
16 gh pr checks 123
```

---

### Actions (workflows & runs)

```
1 gh workflow list
2 gh workflow run ci.yml --ref my-feature-branch
3 gh run list --limit 20
4 gh run watch --exit-status
```

---

### Raw REST API via `gh api`

```
1 # List open issues as JSON (then filter with jq if installed)
2 gh api repos/OWNER/REPO/issues --method GET --paginate \
3   -F state=open -H "Accept: application/vnd.github+json"
```

---

## 11) Repository hygiene

### Recommended `.gitattributes`

---

```
1 # Normalize line endings
2 * text=auto eol=lf
3
4 # Treat these as text
5 *.md text
6 *.yml text
7 *.yaml text
8 *.json text
9
10 # Binary assets (no diff)
11 *.png binary
12 *.jpg binary
13 *.zip binary
```

---

### Useful `README.md` scaffold (no fenced blocks inside)

---

```
1 # Project Name
2
3 Short description. One or two sentences.
4
5 ## Quickstart
6   npm ci
7   npm run dev
8
9 ## Scripts
10 - `npm test` - run test suite
11 - `npm run build` - production build
12
13 ## Contributing
14 See [CONTRIBUTING.md](CONTRIBUTING.md).
15
16 ## License
17 Apache-2.0
```

---

## 12) Troubleshooting

- **PR checks failing:** open the **Checks** tab; re-run & read logs.
- **Auth errors:** `gh auth status`; verify PAT scopes (`repo`, `workflow`).
- **Missing permissions:** ask a maintainer; check branch protection & `CODEOWNERS`.
- **Actions not running:** confirm `on:` triggers and branch filters; check concurrency.
- **Release not visible:** ensure tag exists on origin; use `gh release create`.

*Adapt this sheet with your team's conventions (branch names, required checks, deploy envs). Keep PRs small, reviews fast, and releases frequent.*