GitLab CLI GitLab from the terminal

START HERE

`glab auth login`
Sign in

`glab repo clone <group>/<project>`
Clone a project

`glab mr create --fill`
Open an MR from commits

`glab mr list`
List open MRs

`glab mr checkout <id>`
Check out an MR

`glab mr merge`
Merge the current MR

`glab ci status --live`
Watch the pipeline

`glab ci trace <job>`
Stream a job log

Branch local commits → `glab mr create` → **Merge request** review and approval → `glab mr merge` → **Merged** target branch → `glab ci status` → **Pipeline** CI/CD jobs

CONCEPTS

Inside a clone, glab finds the project from the git remote. Use -R <group>/<project> to target another one.

Merge requests are !123 and issues are #123 on GitLab; glab takes the bare number, a branch name or a URL.

List and view commands accept --output json (-F json) and --jq <expr>. Set GITLAB_HOST to use a self-managed instance for one command.

AUTH AND SETUP

`glab auth login`
Sign in interactively (OAuth or token)

`glab auth status`
Show which account and host you use

`glab auth login --hostname <host>`
Sign in to a self-managed instance

`glab auth login --stdin < <file>`
Sign in with a token from stdin

`glab auth login --device`
Device-flow login on a headless box
Needs GitLab 17.9 or later.

`glab auth status --all`
Check every configured instance

MERGE REQUESTS

`glab mr create --fill`
Open an MR with title from commits

`glab mr create --fill --push --yes`
Push, create, skip all prompts

`glab mr create -t "<title>" -b <branch> --draft`
Draft MR into a target branch

`glab mr create -l <label> --reviewer <user>`
Add labels and a reviewer

`glab mr create -i <issue> --copy-issue-labels`
Create an MR for an issue

`glab mr list`
List open MRs

MERGE REQUESTS (cont.)

`glab mr list --assignee=@me`
MRs assigned to you

`glab mr list --reviewer=@me`
MRs waiting for your review

`glab mr list --all`
Open, closed and merged MRs
Use --all or -A: -a is --assignee.

`glab mr view <id>`
Show an MR

`glab mr view <id> --comments`
Include comments and activity

`glab mr view <id> --web`
Open the MR in the browser

`glab mr checkout <id>`
Check out an MR branch

`glab mr diff <id>`
Show the MR diff

`glab mr approve <id>`
Approve an MR

`glab mr note <id> -m "<text>"`
Comment on an MR

`glab mr merge`
Merge the MR of the current branch

`glab mr merge <id> --squash --remove-source-branch`
Squash-merge and delete the branch

`glab mr merge <id> --auto-merge=false`
Merge now, don't wait for pipeline
With a running pipeline, merge defaults to auto-merge.

`glab mr rebase <id>`
Rebase the source branch

`glab mr update <id> --ready`
Mark a draft ready

`glab mr close <id>`
Close without merging

ISSUES

`glab issue list`
List open issues

`glab issue create -t "<title>" -d "<text>"`
File an issue

`glab issue view <id>`
Show an issue

`glab issue list --assignee=@me`
Issues assigned to you

ISSUES (cont.)

`glab issue list --label <name> --all`
Filter by label, any state

`glab issue create -l <label> --confidential`
Label it and keep it private

`glab issue note <id> -m "<text>"`
Comment on an issue

`glab issue close <id>`
Close an issue

`glab issue update <id> --label <name>`
Change labels and fields

CI/CD

`glab ci status`
Pipeline status of this branch

`glab ci status --live`
Follow the pipeline until it ends

`glab ci status --wait`
Block until done, then print

`glab ci view`
Interactive pipeline view

`glab ci trace <job-name>`
Stream a job's log

`glab ci run`
Start a pipeline on this branch

`glab ci run -b <branch> --variables <KEY>:<value>`
Run with CI variables

`glab ci retry <job-name>`
Retry a job

`glab ci lint`
Validate .gitlab-ci.yml

`glab ci lint --dry-run --include-jobs`
Simulate and list the jobs

`glab ci list`
List recent pipelines

REPOSITORIES

`glab repo clone <group>/<project>`
Clone a project

`glab repo clone -g <group> --paginate`
Clone every project of a group

`glab repo fork --clone`
Fork the current project and clone

`glab repo view --web`
Open the project in the browser

`glab repo create <name>`
Create a project

CI VARIABLES

`glab variable list`
List project CI/CD variables

`glab variable set <KEY> <value>`
Create a variable

`glab variable set <KEY> --masked --protected`
Mask in logs, protected refs only

`glab variable set <KEY> < <file>`
Read the value from a file

API

`glab api projects/:fullpath/releases`
GET an endpoint for this project

`glab api projects/:id/issues --paginate`
Fetch every page

`glab api <endpoint> -f <key>=<value>`
Send a string field (POST)

`glab api <endpoint> -F <key>=<value>`
Typed field: numbers, bools, @file

`glab api -X PUT <endpoint>`
Choose the HTTP method

`glab api graphql -f query='<query>'`
Run a GraphQL query

`glab mr list -F json --jq '.[].title'`
JSON output, filtered with jq

RECIPES

Branch to merged MR

```
git switch -c feature/x
# commit your work
glab mr create --fill --push --yes
glab ci status --live
glab mr merge --squash --remove-source-branch
```

Review someone's MR

```
glab mr list --reviewer=@me
glab mr checkout <id>
glab mr diff
glab mr approve
```

Debug a failed pipeline

```
glab ci status
glab ci trace <job-name>
glab ci retry <job-name>
```

Check CI config and run with a variable

```
glab ci lint
glab ci run -b <branch> --variables DEPLOY_ENV:staging
glab ci status --wait
```