GitLab 活用チートシート

GitHub との違い、Windows と Mac の基礎コマンド、生成AIに MR を任せるときの設定と考え方 ／ 2026-09 時点

Givery

GitLab の位置づけ

ひとことで 計画から CI/CD、デプロイまで1つで持つ DevSecOps 基盤。社内網に丸ごと置けるので閉域の案件向き

提供形態 GitLab.com、Self-Managed（CE は MIT、EE は商用）、Dedicated（単一テナント）

プラン Free、Premium $29（1人月）、Ultimate。Free は非公開グループ5人、CI 月400分

リリース 毎月第3木曜（19.4 は 9/17）。社内の GitLab は版が古く、画面が資料と違いがち

GitHub との対応

Pull Request → Merge Request MR は `!12`、Issue は `#12` で参照する

Organization → グループ 入れ子にでき、権限と設定が下へ継承される

Actions → `.gitlab-ci.yml` `on` → `rules`、`steps` → `script`、`runs-on` → `tags`、`uses` → `include`

Secrets → CI/CD 変数 Masked・Protected・Hidden を選ぶ。yml に鍵を書かない

毎日の git（共通）

鍵 `ssh-keygen -t ed25519` で作り、登録後に `ssh -T git@gitlab.com` で Welcome が出れば OK

ブランチ Issue 番号を名前に入れる

```
git switch -c 123-login-error
```

push と同時に MR

```
git push -u origin HEAD -o merge_request.create
```

公開済みを取り消す 履歴を書き換えない

```
git revert <sha>
```

glab（GitLab CLI）

ログイン 社内 GitLab は `--hostname`

```
glab auth login --hostname gitlab.example.com
```

MR を作る 題名と説明をコミットから埋める

```
glab mr create --fill --draft
```

CI を追う 失敗ログをそのまま AI に渡す

```
glab ci trace
```

MR のコメント `/approve` `/assign_reviewer @GitLabDuo` `/draft` `/ready` が効く

Windows と Mac で違うところ

作業	Windows（PowerShell）	Mac（zsh）
Git と glab を入れる	`winget install --id Git.Git -e`、`winget install glab.glab`（glab はコミュニティ管理）	`brew install git glab`
鍵を ssh-agent に載せる	管理者で `Get-Service ssh-agent | Set-Service -StartupType Automatic`、`Start-Service ssh-agent`、`ssh-add $env:USERPROFILE\.ssh\id_ed25519`	`ssh-add --apple-use-keychain ~/.ssh/id_ed25519`
公開鍵をコピー	`Get-Content $env:USERPROFILE\.ssh\id_ed25519.pub | Set-Clipboard`	`tr -d '\n' < ~/.ssh/id_ed25519.pub | pbcopy`
改行コード	`core.autocrlf true`。`.sh` は `.gitattributes` で `eol=lf` に固定	`core.autocrlf input`
実行ビット	`git update-index --chmod=+x ci/run.sh`	`chmod +x ci/run.sh` してコミット
長いパス	`git config --global core.longpaths true`	設定不要

AI に任せる範囲と GitLab 側の歯止め

任せる作業	AI にさせること	GitLab 側で止める仕組み	人が決めること
MR のレビュー	差分を読み、指摘を JSON に出して MR にコメント1本	読むだけのジョブにする。書き込みはコメントだけ	指摘を採るかどうか
品質ゲート	重大度が閾値以上の指摘が1件でもあれば落とす	最初は `allow_failure: true`。誤検知を見てから外し、Pipelines must succeed で効かせる	閾値
Issue から実装	夜間の schedule で実装し、MR を開く	main は保護して push させない。ボットは Developer まで	差分とテストを見て merge
CI の失敗調査	ジョブログから原因と直し方を出す	`glab ci trace` の出力を渡すか、MR の Fix pipeline with Duo	直し方の採否
MR の説明	テンプレートの影響範囲の表を埋める	`.gitlab/merge_request_templates/Default.md` に欄を用意	影響範囲が正しいか

.gitlab-ci.yml の型

```
ai_review:              # 読むだけ（test ステージ）
  rules:
    - if: $CI_PIPELINE_SOURCE == "merge_request_event"
  script: [bash ci/ai_review.sh]
  artifacts: {when: always, paths: [findings.json]}
ai_gate:                # 閾値以上で落とす
  stage: .post
  rules:
    - if: $CI_PIPELINE_SOURCE == "merge_request_event"
  script: [bash ci/ai_gate.sh findings.json]
  allow_failure: true # 誤検知を見てから外す
nightly:                # 夜間に Issue を実装
  rules:
    - if: $CI_PIPELINE_SOURCE == "schedule"
  timeout: 30m
  variables: {ANTHROPIC_MODEL: claude-sonnet-5-5}
  script:
    - npm i -g @anthropic-ai/claude-code@2.1.284
    - claude -p "$PROMPT" --max-turns 30
```

GitLab Duo

課金は2系統 席課金の Core・Pro・Enterprise と、Credits（1 credit = $1）払いの Agent Platform

MR レビュー `/assign_reviewer @GitLabDuo` で1回 $0.25。設定で自動にもできる。モデルは 8/6 から Claude Sonnet 5

レビューの観点 `.gitlab/duo/mr-review-instructions.yaml` にファイルの種類ごとに書く。足すだけで強制力はない

AGENTS.md は読まない Duo のレビューは AGENTS.md も SKILL.md も見ない。規約は上の yaml へ

外のエージェントをつなぐ

GitLab MCP サーバー MR・Issue・ジョブを読み書きする（Beta、19.2 から Free も）

```
claude mcp add -s user --transport http GitLab https://gitlab.com/api/v4/mcp
```

Claude Code の CI 連携 Beta、保守は GitLab。CI ジョブで `claude -p` を回し、変更は MR にして返す

@claude で呼ぶ 標準では反応しない。Webhook から Trigger API を叩く部分は自前

Codex MR に `@codex review`（Beta）。AGENTS.md の `## Code Review Rules` を読む

先に固める設定

保護ブランチ Settings > Repository > Branch rules。main の push は No one

承認と CODEOWNERS 必須の承認と CODEOWNERS は Premium から

CI/CD 変数 Protected は保護ブランチ同士の MR でしか読めない。鍵は Masked に

ボットの身元 サービスアカウント（席を使わない）。個人のトークンを CI に入れない

考え方とコツ

AI は MR を開くまで merge は人。ボットに Maintainer を渡すと保護ブランチが効かない

人より先に AI GitLab 自身の規程は、人にレビューを頼む前に Duo の指摘へ全部対応すること

版と上限を固定 モデルは正式名、CLI は版まで書く。`--max-turns` と `timeout` は両方

Tips・注意

同じ MR で、上位モデルの全体レビューは6件（重大2件）、下位モデルの差分だけは0件。安いレビューは黙って通す。

夜間実装は16ターンで上限に届き $0.89。`--allowedTools` は `&&` でつないだコマンドを通さない。

フックを CI に移すと、CRLF、実行ビット、絶対パス、コマンドの不在で黙って素通しになる。1回わざと落として確かめる。

Sources（公式・一次情報 / クリックで開く）

1. 料金
2. GitHub Actions からの移行
3. SSH キー
4. ロールと権限
5. MR パイプライン
6. CI/CD 変数
7. glab mr create
8. Code Review Flow
9. レビュー指示
10. GitLab Credits
11. フローの実行
12. GitLab MCP サーバー
13. Claude Code GitLab CI/CD
14. Codex の GitLab 連携
15. GitLab のレビュー規程
16. OpenSSH のキー管理
17. Git の改行コード
18. 資格情報ヘルパー