Engineer

# GitLab

自分のサーバーでも運用できるGit開発基盤

## どのようなものか

GitLabはGitリポジトリーを中心とする一つのアプリです。コードを置き、レビュー用のマージリクエスト、計画用の課題、ビルド、テスト、デプロイを同じ場所から動かすCI/CDを備えます。特徴は実行場所の選択です。GitLab自身が動かすGitLab.com、自社サーバーへ導入するGitLab Self-Managed、一顧客専用のインスタンスを、顧客が選ぶAWSリージョンでGitLabがホストし保守するGitLab Dedicatedがあります。

土台は普通のGitです。各プロジェクトはGitリポジトリーで、通常のコマンドで複製、プッシュ、取得ができ、履歴も完全です。議論、課題、パイプライン結果、権限などの追加部分は、リポジトリーの外でGitLabが保持し、複製には入りません。

マージリクエストは、統合を待つ変更提案で、ほかの基盤のプルリクエストと同じ考え方です。文書は、コードのレビュー、議論、変更の追跡を集める場所と説明します。一ページに、説明、行ごとのレビューコメント付き変更、CI/CDの状態、統合できるかの報告、コミット一覧を示します。課題と結び、マージ時に自動で閉じられます。既定のブランチ、通常mainは初期状態で保護され、Maintainer以上の役割だけがマージできます。

CI/CDは最上位の `.gitlab-ci.yml` で設定します。パイプラインを作る段階、ジョブ、スクリプトを宣言します。段階はビルド、テスト、デプロイなどの順番、ジョブは各段階の仕事です。実行するGitLab Runnerは、Go製の別プログラムで、単一バイナリーとして配布します。GitLabへ接続してジョブを待ち、ランナー自身のマシン、Dockerコンテナーの内部、またはSSH先のリモートサーバーで動かし、結果を返します。

概念を判断に生かす

「GitLabとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 変更を検証し、リリースし、元に戻せる仕組みを整え、金曜の深夜ではなく火曜の午後にお客様へ届けられるようにします。

**サービスを見る** お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 GitHub ActionsでGIS SaaSのデプロイを自動化 — デプロイ、データ、報告を手作業から解放

**GitHub ActionsでGIS SaaSのデプロイを自動化** お問い合わせ →

## なぜ重要なのか

コードを書いてから配布するまでの大部分を、一つのGitLab内で行えます。コード、レビュー、テストするパイプライン、依頼の課題、誰が何を承認したかの記録がそろいます。アプリが一つなら、アカウントと権限も一組、検索場所も一か所で、課題、それに応えたマージリクエスト、その変更を検査したパイプラインのつながりも保てます。

自社運用できることが、選ぶ理由になる組織も多くあります。契約、規制、社内方針でコードを外へ出せないなら、外部サービスは使えません。GitLabは、保管を求められるネットワーク内へ導入できます。公式文書も正式な用途として扱い、Linuxパッケージ、KubernetesのHelmチャートとともに、1,000人から50,000人向けの参照構成を公開します。

その制御には代償があり、ライセンス料だけでなく運用作業も必要になります。適切なサーバーを用意し、PostgreSQLとRedisまたはValkeyを健全に保ち、ランナーのマシンを用意し、実際に復元できるバックアップを取り、指定順で更新します。運用込みの選択肢ならGitLabに任せますが、自社運用では引き受けます。

## 実際の活用例

自社の仕事は、GitLabの両面に触れています。

数年のデータエンジニアリングでは、GitHubとGitLabのCI/CDと、その下のBash、Pythonが、分析と開発の手作業を置き換えました。開発から提供までを標準化し、案件ごとに作り直さず同じ方法にしました。別世界として手作業が多かった分析も、開発と同じパイプラインへ入りました。事例は約4,000時間の削減と、毎回少し違うやり方をしなくなったことで、より一貫した提供につながったと記録します。

自社コードは、自前ホスティングの対照的な選択を示します。第三者サービスから移した先は、単一バイナリーで、SSH中心、管理用ウェブ画面を持たない自社フォージです。インターフェースが少なければ守る対象も少ないという原則で選びました。これは、使用可能量が約464 MBの、メモリー512 MBの公開サービス用ホストを、七ドメインのウェブサーバー、Torのonionサービス、Geminiサーバー、Gopherサーバー、不正アクセス元の遮断と共有します。バックアップの自動化は別に実装していますが、実装と稼働中のバックアップは別です。

どちらの事例もGitLabには言及せず、次は本稿独自の計算です。GitLabが示す単一ノードの最低メモリーは8 GBで、そのホスト全体は十六分の一です。自分でGitLabを運用できるとは、それに合うサーバーを持つことです。小さく単機能のフォージも、自分のマシンにコードを置く別の方法です。

順に実行する二つの段階に、それぞれ一つのジョブを置いた最小限のパイプラインファイルで、CI/CD設定の形を示します。

```
stages:
    - build
    - test

build-site:
    stage: build
    script:
        - make build

run-tests:
    stage: test
    script:
        - make test
```

## 問題が起きるところ

- **小さいサーバーで全体が遅くなります。** 公式文書は単一ノードの基準を16 GB、メモリー制約のある環境で動く最低を8 GBとします。負荷時のスワップは性能を大きく落とす可能性があると警告し、可能なら無効にするか、使わず済む十分なメモリーを求めます。
- **復元したGitLabが自身のデータを読めません。** バックアップコマンドは `/etc/gitlab` を含めず、その中の `gitlab-secrets.json` も除きます。データベースの暗号化した値を開く鍵で、なければ復元後も読めません。設定は別に、最低限 `gitlab-secrets.json` と `gitlab.rb` をバックアップするよう求めます。

- **必須の版を飛ばして更新します。** 更新経路には、後の版へ進む前に必ず入れる中継点があります。GitLab 17.5以降はx.2、x.5、x.8、x.11です。各中継点のバックグラウンド移行が終わるまで次へ進めず、何版も遅れたものは一点ずつ追い付きます。
- **管理サービスの保存先が非対応です。** 対応するデータベースはPostgreSQLだけです。セッションとバックグラウンド処理のRedisかValkeyは独立インスタンスが必要で、Redis Cluster、RedisやValkeyのサーバーレス版は非対応です。クラウドの管理サービスでも、対応する独立型なら使えます。
- **想定しなかった実行マシンを待ちます。** GitLab.comには用意済みのランナーがありますが、自社運用では管理者がCI/CDの基盤を提供、管理し、組織の負荷に足りる容量も確保します。忙しいチームが小さいランナー一台を共有すると、列を作って待ちます。

## 参考資料

- GitLab Docs — 利用、導入、管理に関する包括的なドキュメントです。
- 導入要件 — メモリー、CPU、保存領域、対応するPostgreSQLとRedisの版です。
- GitLabのバックアップ — 含むもの、除くもの、機密情報を含む設定を、バックアップとともに保存する方法です。
- 更新経路 — 必須の中継点と、その間でバックグラウンド移行を終える理由です。
- GitLab CI/CD入門 — `.gitlab-ci.yml` 、段階、ジョブ、パイプラインです。
- GitLab Runner — ジョブを動かすプログラムと、その実行方法です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

**料金を見る** お問い合わせ →

# 用語を実務に結び付ける

「GitLabとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/gitlab/