**GitLab**

GitWeb धेरै साधारण छ। यदि तपाईं आधुनिक, पूर्ण सुविधायुक्त Git सर्भर खोज्दै हुनुहुन्छ भने, त्यहाँ धेरै ओपन-सोर्स समाधानहरू उपलब्ध छन् जसलाई तपाईं स्थापना गर्न सक्नुहुन्छ। GitLab तीमध्ये एक लोकप्रिय विकल्प हो, त्यसैले हामी यसलाई स्थापना र प्रयोग गर्ने प्रक्रिया उदाहरणको रूपमा समावेश गर्नेछौं।  
यसलाई GitWeb भन्दा स्थापना गर्न गाह्रो छ र अधिक मर्मत आवश्यक पर्न सक्छ, तर यो पूर्ण सुविधायुक्त विकल्प हो।

**स्थापना ( INSTALLATION )**

GitLab एक डेटाबेस-आधारित वेब अनुप्रयोग हो, त्यसैले यसको स्थापना अन्य केही Git सर्भरहरूको तुलनामा अलि जटिल छ। सौभाग्यवश, यो प्रक्रिया राम्रोसँग दस्तावेजीकृत र समर्थित छ। GitLab ले आधिकारिक Omnibus GitLab प्याकेज मार्फत तपाईंको सर्भरमा GitLab स्थापना गर्न कडा सिफारिस गर्छ।

अन्य स्थापना विकल्पहरू निम्नानुसार छन्:

* **GitLab Helm चार्ट**, जुन Kubernetes सँग प्रयोग गर्नको लागि हो।
* **Dockerized GitLab प्याकेजहरू**, जुन Docker सँग प्रयोग गर्न सकिन्छ।
* **स्रोत फाइलहरूबाट स्थापना।**
* **AWS, Google Cloud Platform, Azure, OpenShift, र Digital Ocean जस्ता क्लाउड प्रदायकहरू।**

**प्रशासन (ADMINSTRATION)**

GitLab को प्रशासन इन्टरफेस वेब मार्फत पहुँच गर्न सकिन्छ। तपाईंको ब्राउजरलाई GitLab स्थापना गरिएको होस्टनेम वा IP ठेगानामा सोझै संकेत गर्नुहोस् र **root** प्रयोगकर्ताको रूपमा लगइन गर्नुहोस्। पासवर्ड तपाईंको स्थापना प्रकारमा निर्भर गर्दछ, तर पूर्वनिर्धारित रूपमा, **Omnibus GitLab** ले स्वचालित रूपमा पासवर्ड उत्पन्न गर्दछ र कम्तिमा 24 घण्टासम्मका लागि यसलाई **/etc/gitlab/initial\_root\_password** फाइलमा भण्डारण गर्छ। थप विवरणहरूका लागि आधिकारिक दस्तावेजीकरण अनुसरण गर्नुहोस्।

लगइन गरेपछि, माथिल्लो दायाँ कुनामा रहेको मेनुमा **"Admin area"** आइकनमा क्लिक गर्नुहोस्।

A screenshot of a computer

Description automatically generated

**प्रयोगकर्ताहरू (USER)**

तपाईंको GitLab सर्भर प्रयोग गर्ने प्रत्येक व्यक्तिसँग एउटा प्रयोगकर्ता खाता (user account) हुनु आवश्यक छ। प्रयोगकर्ता खाता सरल हुन्छ, जसमा मुख्य रूपमा लगइन डाटासँग संलग्न व्यक्तिगत जानकारी समाविष्ट हुन्छ।

प्रत्येक प्रयोगकर्ताको खातामा एउटा **namespace** हुन्छ, जुन उक्त प्रयोगकर्तासँग सम्बन्धित परियोजनाहरू (projects) को तार्किक समुह हो। उदाहरणका लागि, यदि **jane** नामक प्रयोगकर्ताले **project** नामक परियोजना बनाएको छ भने, त्यस परियोजनाको URL निम्नानुसार हुनेछः

http://server/jane/project

A screenshot of a computer

Description automatically generated

GitLab मा प्रयोगकर्ता खाता हटाउने दुई तरिकाहरू छन्:

1. **प्रयोगकर्ता "ब्लक" गर्ने (Blocking):**
   * प्रयोगकर्तालाई ब्लक गर्दा, उनीहरूले GitLab मा लगइन गर्न सक्दैनन्।
   * तर, उक्त प्रयोगकर्ताको **namespace** भित्रको सबै डाटा सुरक्षित रहन्छ।
   * सो प्रयोगकर्ताको ईमेल ठेगाना प्रयोग गरी गरिएको commits हरू अझै पनि उनीहरूको प्रोफाइलमा लिंक गरिनेछ।
2. **प्रयोगकर्ता "नष्ट" गर्ने (Destroying):**
   * प्रयोगकर्तालाई नष्ट गर्दा, उनीहरूको सम्पूर्ण जानकारी डेटाबेस र फाइल प्रणालीबाट हटाइन्छ।
   * प्रयोगकर्ताको **namespace** अन्तर्गत रहेका सबै परियोजनाहरू र डाटाहरू मेटाइन्छ।
   * उनीहरूले स्वामित्व गरेका कुनै पनि समूहहरू (groups) पनि स्थायी रूपमा मेटाइन्छ।
   * यो अत्यन्त स्थायी र विनाशकारी प्रक्रिया हो, जसको आवश्यकता निकै कम मात्र पर्न सक्छ।

**समूहहरू (Groups)**

GitLab मा समूह भनेको परियोजनाहरूको संग्रह हो, जसमा ती परियोजनाहरूमा प्रयोगकर्ताहरूले कसरी पहुँच पाउने भन्ने जानकारी समावेश हुन्छ। प्रत्येक समूहसँग पनि **परियोजना namespace** हुन्छ, जसरी प्रयोगकर्ताहरूको हुन्छ।

उदाहरणका लागि, यदि **training** नामक समूहमा **materials** परियोजना छ भने, त्यस परियोजनाको URL यस प्रकार हुनेछ:  
http://server/training/materials

**समूहको विशेषताहरू**

1. **प्रयोगकर्ताहरू र तिनीहरूको अनुमति (Permissions):**
   * प्रत्येक समूहमा विभिन्न प्रयोगकर्ताहरू आवद्ध हुन सक्छन्, र तिनीहरूलाई समूह तथा परियोजनाहरूमा विभिन्न तहको अनुमति दिइन्छ।
   * अनुमतिहरू साधारण "Guest" (केवल समस्याहरू (issues) र च्याट गर्न) बाट "Owner" (समूह, सदस्य र परियोजनाहरूको पूर्ण नियन्त्रण) सम्म फरक-फरक हुन सक्छ।
   * GitLab ले प्रशासन स्क्रीनमा विभिन्न अनुमतिहरूको विस्तृत सूची प्रदान गर्दछ।

**समूहको प्रयोगका फाइदाहरू**

* परियोजनाहरूलाई व्यवस्थित गर्न मद्दत गर्छ।
* विभिन्न स्तरका प्रयोगकर्ताहरूलाई उचित अनुमति प्रदान गर्न सजिलो बनाउँछ।
* परियोजनाहरूको व्यवस्थापनलाई केन्द्रीय रूपमा नियन्त्रण गर्न सक्षम बनाउँछ।

A screenshot of a web page

Description automatically generated

**परियोजनाहरू (Projects)**

GitLab मा एउटा परियोजना (Project) सामान्य रूपमा एउटा Git भण्डारण (repository) सँग सम्बन्धित हुन्छ। प्रत्येक परियोजना **एउटा namespace** (प्रयोगकर्ता वा समूह) अन्तर्गत पर्दछ।

* यदि परियोजना कुनै **प्रयोगकर्ताको** नाममा छ भने, त्यस प्रयोगकर्ताले सो परियोजनामा पहुँच व्यवस्थापनको प्रत्यक्ष नियन्त्रण गर्छ।
* यदि परियोजना कुनै **समूहको** नाममा छ भने, त्यस समूहमा दिइएको अनुमति स्तर (permissions) परियोजनामा लागू हुन्छ।

**परियोजनाको दृश्यता स्तर (Visibility Levels)**

प्रत्येक परियोजनाले आफ्नै **दृश्यता स्तर (visibility level)** निर्धारण गर्दछ, जसले कोही प्रयोगकर्ताले परियोजनाका पृष्ठहरू र भण्डारण (repository) हेर्न सक्ने अधिकार व्यवस्थापन गर्छ।

1. **Private (निजी):**
   * परियोजनाका मालिकले विशेष प्रयोगकर्ताहरूलाई मात्र पहुँच अनुमति दिन सक्दछन्।
   * अरू कुनै प्रयोगकर्ताले अनुमति बिना परियोजनालाई हेर्न सक्दैन।
2. **Internal (आन्तरिक):**
   * परियोजना कुनै पनि **लग-इन** (logged-in) प्रयोगकर्ताले हेर्न सक्छन्।
   * सार्वजनिक रूपमा बाहिरी प्रयोगकर्ताले पहुँच पाउँदैनन्।
3. **Public (सार्वजनिक):**
   * परियोजना कुनै पनि प्रयोगकर्ताले (लग-इन बिना नै) हेर्न सक्छन्।
   * सार्वजनिक पहुँचका लागि उपयुक्त।

**महत्त्वपूर्ण कुरा:**  
परियोजनाको दृश्यता सेटिङले **Git fetch** मार्फत पहुँचको साथै वेब इन्टरफेस (web UI) पहुँचलाई पनि नियन्त्रण गर्छ।

**हुकहरू (Hooks)**

GitLab ले **हुकहरू (Hooks)** को समर्थन प्रदान गर्छ, जसलाई **परियोजना (Project)** वा **प्रणाली (System)** स्तरमा सेटअप गर्न सकिन्छ। हुकहरू Git भण्डारणलाई अन्य विकास प्रक्रिया (development automation) जस्तै **CI servers, chat rooms, deployment tools** आदिसँग जडान गर्न महत्वपूर्ण हुन्छ।

**हुकहरूको कार्यप्रणाली**

जब GitLab मा कुनै विशेष घटना (event) घट्छ, तब GitLab सर्वरले **HTTP POST अनुरोध** मार्फत **JSON ढाँचामा जानकारी** पठाउँछ। यो जानकारी हुकको माध्यमबाट अन्य सिस्टमहरूसँग शेयर गर्न सकिन्छ।

**हुकहरूको प्रकार**

GitLab मा दुई प्रकारका हुकहरू समर्थित छन्:

1. **परियोजना-स्तर हुक (Project-Level Hooks):**
   * कुनै विशिष्ट परियोजनाको घटनाहरूमा प्रतिक्रिया जनाउने।
   * जस्तै: जब कोड पुश गरिन्छ, मर्ज अनुरोध (merge request) सिर्जना गरिन्छ, वा कुनै मुद्दा (issue) बन्द हुन्छ।
   * परियोजना सेटिङहरू (Project Settings) बाट कन्फिगर गर्न सकिन्छ।
2. **प्रणाली-स्तर हुक (System-Level Hooks):**
   * सम्पूर्ण GitLab प्रणाली भरि घट्ने घटनाहरूमा प्रतिक्रिया जनाउने।
   * सबै परियोजनाहरू र प्रयोगकर्ताहरूका लागि लागू हुन्छ।
   * GitLab प्रशासन प्यानलबाट व्यवस्थापन गरिन्छ।

**हुकहरूको उपयोग**

GitLab हुकहरूको प्रयोग निम्नका लागि गर्न सकिन्छ:

* **निरन्तर एकीकरण (Continuous Integration - CI):** कोड पुश हुँदा CI/Build प्रक्रिया ट्रिगर गर्न।
* **सूचना प्रणाली (Notifications):** च्याट एप्लिकेसनहरू (जस्तै Slack, Microsoft Teams) मा स्वचालित सूचना पठाउन।
* **स्वचालित परिनियोजन (Deployment):** नयाँ कोड अपडेट भएपछि स्वचालित रूपमा उत्पादन वातावरणमा तैनाथ (deploy) गर्न।

**हुकहरू सेटअप गर्ने तरिका**

परियोजना स्तरको हुक सेटअप गर्न:

1. आफ्नो परियोजना खोल्नुहोस्।
2. **Settings > Webhooks** मा जानुहोस्।
3. लक्षित URL प्रविष्ट गर्नुहोस् (जहाँ POST अनुरोध जान्छ)।
4. घटनाहरू (events) चयन गर्नुहोस् जस्तै Push, Merge, Issue आदि।
5. "Add Webhook" क्लिक गर्नुहोस्।

हुकहरू GitLab मा अन्य उपकरणहरूसँग जडान गर्न सहज र उपयोगी तरीका हो।

**बुनियादी उपयोग (Basic Usage)**

GitLab मा काम सुरु गर्नको लागि **नयाँ परियोजना (New Project)** सिर्जना गर्नु महत्त्वपूर्ण छ।

**नयाँ परियोजना बनाउने तरिका**

1. **"+ आइकन"** मा क्लिक गर्नुहोस्।
2. निम्न जानकारीहरू प्रविष्ट गर्नुहोस्:
   * **परियोजनाको नाम (Project Name)**
   * **यसले कुन नामस्थान (Namespace) अन्तर्गत पर्नु पर्छ।**
   * **दृश्यता स्तर (Visibility Level):** परियोजना निजी (Private), आन्तरिक (Internal), वा सार्वजनिक (Public) बनाउने विकल्प।
3. सेटिङ इन्टरफेसमार्फत पछि यी विवरणहरू परिवर्तन गर्न सकिन्छ।
4. "Create Project" क्लिक गरेपछि तपाईंको परियोजना तयार हुन्छ।

**स्थानीय Git भण्डारण (Local Git Repository) जडान गर्ने**

नयाँ परियोजना बनाएपछि, तपाईंले आफ्नो **स्थानीय Git भण्डारण (Local Repository)** परियोजनासँग जडान गर्न चाहनुहुन्छ। प्रत्येक परियोजना **HTTPS वा SSH** मार्फत पहुँचयोग्य हुन्छ।

**अवस्थित स्थानीय भण्डारण (Existing Local Repository) जडान गर्न:**

तलको आदेश (command) चलाउनुहोस्:

git remote add gitlab https://server/namespace/project.git

**नयाँ क्लोन (Clone) गर्न:**

यदि तपाईं सँग कुनै स्थानीय भण्डारण छैन भने, तलको आदेश प्रयोग गर्नुहोस्:

git clone https://server/namespace/project.git

**GitLab वेब इन्टरफेस**

GitLab को वेब UI ले परियोजना भण्डारणको विभिन्न उपयोगी दृश्यहरू प्रदान गर्छ।

* **होम पृष्ठ (Home Page):**
  + हालैका गतिविधिहरू (Recent Activity) देख्न सकिन्छ।
  + शीर्ष भागमा रहेका लिंकहरूबाट फाइलहरू (Files), कमिट लग (Commit Log), र अन्य सेटिङहरू हेर्न सकिन्छ।

यस प्रक्रियाबाट तपाईंले आफ्नो परियोजनालाई GitLab मा सहजै व्यवस्थापन गर्न सक्नुहुन्छ।

**साथमा काम गर्ने तरिका (Working Together)**

GitLab मा परियोजनाहरूमा सहकार्य गर्नका लागि विभिन्न तरिकाहरू छन्, जसले टीमलाई प्रभावकारी रूपमा मिलेर काम गर्न मद्दत गर्छ।

**प्रत्यक्ष पहुँच (Direct Access)**

सबैभन्दा सरल तरीका भनेको **प्रत्येक प्रयोगकर्तालाई प्रत्यक्ष Git भण्डारणमा पुश (push) गर्ने अनुमति दिनु** हो।  
तपाईं कुनै परियोजनामा प्रयोगकर्तालाई थप्नका लागि, सो परियोजनाको **"Members" खण्ड** मा जानुपर्छ।  
त्यहाँ, नयाँ प्रयोगकर्तालाई निश्चित पहुँच स्तर (Access Level) तोक्न सकिन्छ।

उदाहरणका लागि:

* यदि प्रयोगकर्तालाई "Developer" वा सोभन्दा उच्च स्तरको अनुमति दिइयो भने, उसले सोझै कोड पुश गर्न सक्छ।
* "Reporter" वा सोभन्दा तल्लो स्तरका प्रयोगकर्ताहरूलाई पढ्ने अनुमति मात्र प्रदान गरिन्छ।  
  यस विधिले परियोजनामा **टिम वर्कलाई सिधै समन्वय गर्न** सजिलो बनाउँछ।

**मर्ज अनुरोध (Merge Requests) मार्फत सहकार्य**

प्रत्यक्ष पहुँच दिने तरिका बाहेक, **मर्ज अनुरोध (Merge Requests)** एक प्रभावकारी तरिका हो जसले परियोजनामा योगदानको एक नियन्त्रित माध्यम प्रदान गर्छ।

यो विधिमा:

* परियोजनालाई हेर्न सक्ने कुनै पनि प्रयोगकर्ताले आफ्नो परिवर्तनहरू प्रस्ताव गर्न सक्छ।
* जसलाई प्रत्यक्ष पहुँच छ, उनीहरूले नयाँ शाखा सिर्जना गरी कोड पुश गरेर मर्ज अनुरोध खोल्न सक्छन्।
* जसलाई प्रत्यक्ष पहुँच छैन, उनीहरूले परियोजनालाई **फोर्क (Fork)** गरेर आफ्नो प्रति (copy) बनाउनेछन्। त्यसपछि आफ्नै भण्डारणमा परिवर्तन गरी मर्ज अनुरोध पठाउन सकिन्छ।

यस विधिले परियोजना मालिकलाई **कुन योगदानहरू परियोजनामा समावेश गर्ने भन्ने पूर्ण नियन्त्रण** दिन्छ, जबकि बाहिरी र अविश्वसनीय प्रयोगकर्ताहरूलाई पनि योगदान गर्ने अवसर दिन्छ।

**कोड समीक्षा र छलफल (Code Review & Discussion)**

GitLab मा मर्ज अनुरोधहरू र मुद्दाहरू (issues) लामो समयसम्म चल्ने छलफलका प्रमुख एकाइहरू हुन्।

* मर्ज अनुरोधमा प्रस्तावित परिवर्तनहरू लाइन-लाइन समीक्षाको लागि स्थान दिइन्छ।
* यसले **हल्का कोड समीक्षा प्रक्रिया** सपोर्ट गर्छ, जसमा कुनै पनि परिवर्तनमा विस्तृत छलफल गर्न सकिन्छ।
* प्रत्येक मर्ज अनुरोध र मुद्दाहरू निश्चित प्रयोगकर्तालाई **"assign"** गर्न सकिन्छ, र निश्चित समयसीमा (milestones) अनुसार व्यवस्थापन गर्न सकिन्छ।

**अन्य सहकार्य सुविधाहरू (Additional Collaboration Features)**

GitLab को Git सम्बन्धित सुविधाहरू मात्र नभई अन्य सहकार्यका उपकरणहरू पनि उपलब्ध छन्, जस्तै:

* **परियोजना विकीहरू (Project Wikis):**
  + परियोजनासँग सम्बन्धित महत्वपूर्ण जानकारी सुरक्षित राख्न।
* **सिस्टम व्यवस्थापन उपकरणहरू (System Maintenance Tools):**
  + GitLab प्रशासकहरूका लागि व्यवस्थापन सजिलो बनाउन।

GitLab प्रयोग गर्दा, एकचोटि सर्वर सेटअप गरेपछि अधिकांश प्रशासनिक कार्यहरू **वेब इन्टरफेस** मार्फत सहजै गर्न सकिन्छ। SSH वा कन्फिगरेशन फाइलहरू परिमार्जन गर्नुपर्ने आवश्यकता प्रायः हुँदैन।

यसरी, GitLab मा प्रत्यक्ष पहुँच, मर्ज अनुरोधहरू, कोड समीक्षाहरू, र सहकार्यका विभिन्न सुविधाहरू प्रयोग गरेर कुनै पनि टीमले सहज, सुरक्षित र व्यवस्थित रूपमा परियोजनाहरू सञ्चालन गर्न सक्छ।

# Third Party Hosted Options

यदि तपाईंले आफ्नो Git सर्भर सेटअप गर्न चाहनुहुन्न भने, तपाईंले बाह्य समर्पित होस्टिङ साइटमा Git परियोजनाहरू होस्ट गर्ने थुप्रै विकल्पहरू छनोट गर्न सक्नुहुन्छ। यसले केहि फाइदाहरू प्रदान गर्दछ: एक होस्टिङ साइट सामान्यतया छिटो सेटअप गर्न योग्य र परियोजनाहरू सुरु गर्न सजिलो हुन्छ, र कुनै सर्भर मर्मत वा निगरानीको आवश्यकता हुँदैन। यदि तपाईंले आन्तरिक रूपमा आफ्नो सर्भर सेटअप र चलाउनुभयो भने पनि, तपाईंले सार्वजनिक होस्टिङ साइट प्रयोग गर्न चाहनुहुन्छ भने — यो खुला स्रोत समुदायलाई तपाईंको कोड सजिलै फेला पार्न र तपाईंलाई मद्दत गर्न सजिलो हुन्छ।

आजकाल, तपाईंलाई चयन गर्नका लागि थुप्रै होस्टिङ विकल्पहरू उपलब्ध छन्, प्रत्येकसँग फरक फरक फाइदाहरू र घाटाहरू छन्। यद्यपि यसको बारेमा अपडेटेड सूची हेर्नको लागि, तपाईं GitHosting पृष्ठमा जान सक्नुहुन्छ जुन मुख्य Git विकीमा उपलब्ध छ: <https://archive.kernel.org/oldwiki/git.wiki.kernel.org/index.php/GitHosting.html>

यस लेखमा हामी GitHub को उपयोग बारे विस्तृत रूपमा छलफल गर्नेछौं, किनकि यो सबैभन्दा ठूलो Git होस्ट हो र तपाईंलाई यसको साथ काम गर्नका लागि यससँग अन्तर्क्रिया गर्नुपर्ने हुन सक्छ। तर तपाईंको आवश्यकता र चाहनाअनुसार, तपाईंले अन्य थुप्रै होस्टिङ प्लेटफर्महरू छनोट गर्न सक्नुहुन्छ यदि तपाईं आफ्नै Git सर्भर सेटअप गर्न चाहनुहुन्न भने।

**Git Hosting को फाइदाहरू:**

1. **छिटो सेटअप र सुरु गर्न सजिलो**: बाह्य होस्टिङ साइटको उपयोग गर्दा, तपाईंले आफ्नो Git सर्भर सेटअप गर्नको लागि धेरै समय र प्रयास बचत गर्न सक्नुहुन्छ।
2. **सर्भर मर्मत र निगरानी बिना**: तपाईंले सर्भरको मर्मत गर्न र निगरानी राख्नुपर्ने आवश्यकता छैन, जसले तपाईंलाई परियोजनामा बढी ध्यान केन्द्रित गर्न मद्दत पुर्याउँछ।
3. **खुला स्रोत समुदायसँग सजिलै सहयोग**: सार्वजनिक होस्टिङ साइटहरूमा तपाईंको परियोजनाहरू राख्दा, खुला स्रोत समुदायको समर्थन प्राप्त गर्न सजिलो हुन्छ।

**GitHub र अन्य होस्टिङ प्लेटफर्महरू:**

GitHub सबैभन्दा ठूलो Git होस्ट हो, र यसमा धेरै उपकरणहरू र सुविधाहरू उपलब्ध छन् जसले तपाईंलाई कन्फिगर र मर्मतको झन्झट बिना काम गर्न मद्दत गर्छ। यद्यपि, तपाईंले अन्य थुप्रै होस्टिङ विकल्पहरू पनि प्रयोग गर्न सक्नुहुन्छ जुन तपाईंको परियोजनाको आवश्यकतासँग मेल खान्छ।

-------------------------------------------------------------END-------------------------------------------------------------