Aembit Server Workload Cookbook: *GitLab Edition*

How to Securely Configure Workload Access to the GitLab REST API

GitLab has become a central platform for software delivery across industries. As of 2025, it supports more than 50 million registered users globally, with over half of the Fortune 100 relying on GitLab to automate development, operations, and security workflows.

GitLab's growth reflects both the ubiquity of CI/CD adoption – and the increasing complexity of managing authentication between automated systems, applications, and external services securely and at scale.

The GitLab REST API sits at the center of these interactions, which provides programmatic access to GitLab's key functionality – including repository management, merge requests, and issue tracking. Workloads, such as applications, AI agents, and services, frequently interact with the REST API as part of broader delivery and orchestration pipelines. These workloads require authentication to perform essential tasks within automated environments.

However, the prevailing methods for managing this access introduce unnecessary operational and security challenges. It is common for organizations to store static personal access tokens or application secrets as environment variables or configuration files, making them difficult to rotate, audit, and protect with fine-grained access controls – and exposing organizations to risk when mishandled.

The consequences of such practices have become apparent in recent breaches. The 2025 Verizon Data Breach Investigations Report found that stolen credentials remain the most frequent initial attack vector, contributing to roughly a quarter of breaches. Specific to the DevOps environment, the breach of Pearson plc in January 2025 began with a leaked GitLab token, leading to the exfiltration of terabytes of sensitive data. These incidents illustrate how a single mismanaged credential can escalate into large-scale compromise.

In response to these realities, this edition of the **Aembit Server Workload Cookbook** offers clear, technically rigorous "recipes" for securely connecting workloads to the GitLab REST API. The guide details practical procedures for establishing authentication via OAuth 2.0 Authorization Code flows, along with recommended practices for reducing reliance on long-lived secrets and improving auditability.

This guidance is intended to be relevant regardless if your organization uses the Aembit Workload IAM Platform to secure workload access. The patterns described here apply equally to teams leveraging GitLab's native security features, cloud identity providers, or other secret management frameworks.

Readers will find the following material:

- Reference architectures illustrating authentication patterns suitable for machine workloads accessing GitLab's REST API.
- Detailed configuration instructions for OAuth 2.0 Authorization Code flow, emphasizing correct implementation and adaptability to enterprise requirements.
- Considerations for runtime behavior, such as how libraries and tools interact with injected credentials.
- Recommendations for minimizing credential exposure, including policy enforcement, centralized credential lifecycle management, and support for short-lived tokens.
- Design principles applicable across varied infrastructure environments, including on-premises, hybrid, and cloud-native deployments.

This publication is intended for DevOps engineers, platform architects, and security professionals responsible for maintaining the integrity of workload-to-service interactions. The objective is to support informed decision-making and consistent execution in the secure management of workload credentials within the context of GitLab and related systems.

Future editions of this series will extend similar guidance to additional platforms and services, reflecting a broader effort to advance the security and efficiency of machine identity management throughout the enterprise.

Now let’s get back to the kitchen to do some cooking!

Prerequisites

Before proceeding with the configuration, you must have a GitLab tenant (or sign up for one) and a user, group, or instance level owned application. If you have not generated an application yet, follow the configuration steps below. For detailed information on how to create a new application, please refer to the official GitLab documentation.

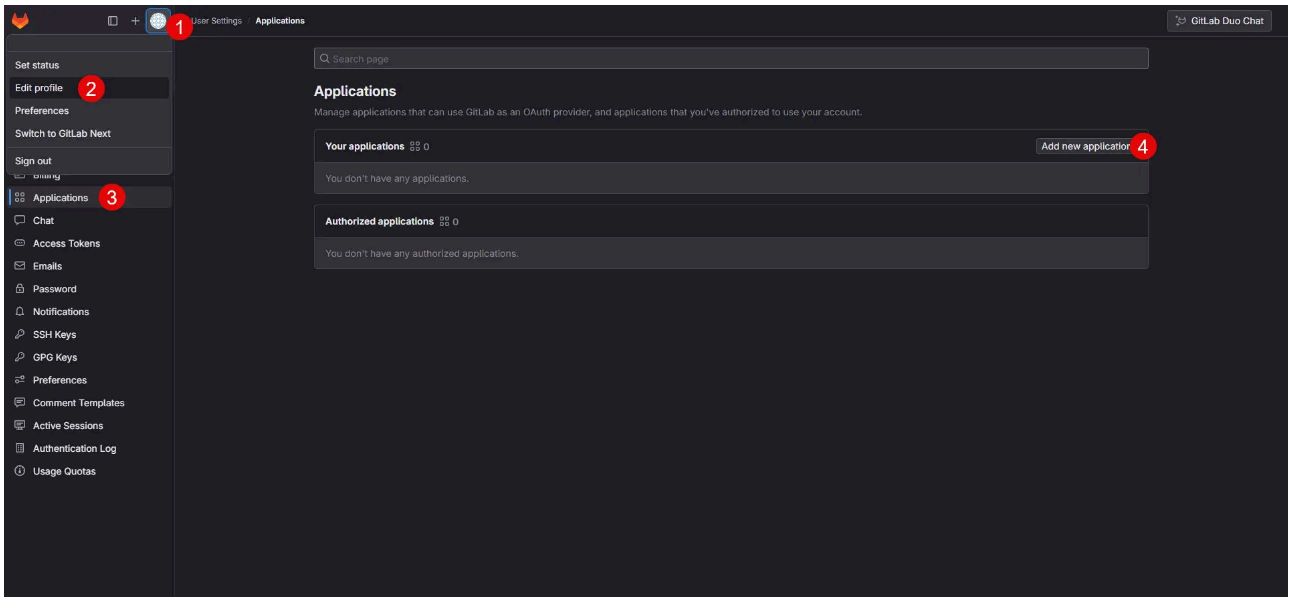
Server Workload Configuration

1. Create a new Server Workload.
   - **Name** - Choose a user-friendly name.

2. Configure the service endpoint:
   - **Host** - gitlab.com
   - **Application Protocol** - HTTP
   - **Port** - 443 with TLS
   - **Forward to Port** - 443 with TLS
   - **Authentication method** - HTTP Authentication
   - **Authentication scheme** - Bearer

Credential Provider Configuration

1. Sign in to your GitLab account.
2. In the upper-left corner of any page, click your profile photo, then click **Edit Profile.**
3. Navigate to **Applications** in the left-hand menu.
4. On the right side, click on the **Add new application** button.



Applications

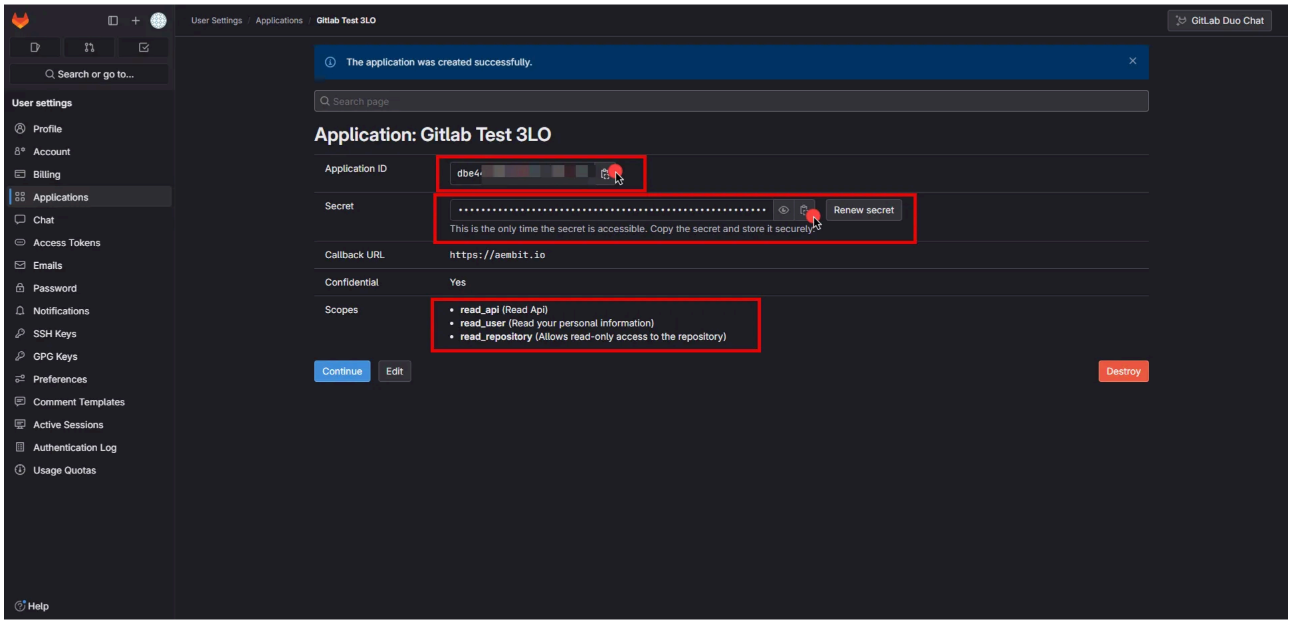
Manage applications that can use GitLab as an OAuth provider, and applications that you've authorized to use your account.

You don't have any applications.

You don't have any authorized applications.

5. Provide a name for your app.
6. Switch to the Aembit UI to create a new Credential Provider, selecting the OAuth 2.0 Authorization Code credential type. After setting up the Credential Provider, copy the auto-generated Callback URL.
7. Return to GitLab and paste the copied URL into the **Redirect URI** field.
8. Check the **Confidential** box, and select the scopes for your application depending on your needs.
9. After making all of your selections, click on **Save application**.

10. On the directed page, copy the **Application ID, Secret** and **Scopes**, and store them for later use in the tenant configuration.



The application was created successfully.

Application: Gitlab Test 3LO

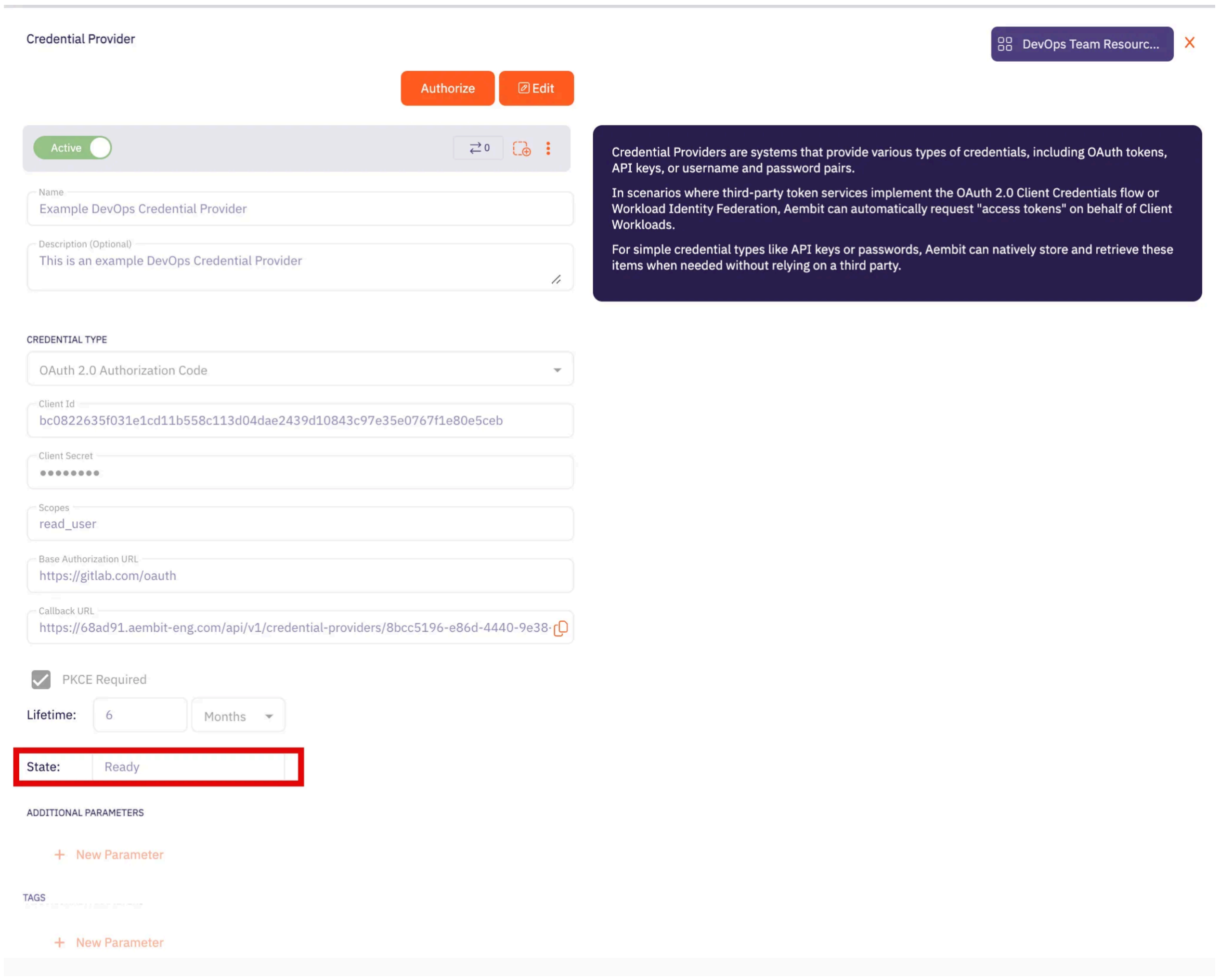
Application ID	dbe4[redacted]
Secret	•• This is the only time the secret is accessible. Copy the secret and store it securely.
Callback URL	https://aembit.io
Confidential	Yes
Scopes	• **read_api** (Read Api) • **read_user** (Read your personal information) • **read_repository** (Allows read-only access to the repository)

11. Edit the existing Credential Provider created in the previous steps.

- **Name** - Choose a user-friendly name.
- **Credential Type** - OAuth 2.0 Authorization Code
- **Callback URL (Read-Only)** - An auto-generated Callback URL from Aembit Admin.
- **Client Id** - Provide the Application ID copied from GitLab.
- **Client Secret** - Provide the Secret copied from GitLab.
- **Scopes** - Enter the scopes you use, space-delimited (e.g. `read_api` `read_user` `read_repository`).
- **OAuth URL** - https://gitlab.com
- Click on **URL Discovery** to populate the Authorization and Token URL fields, which can be left as populated.
- **PKCE Required** - On
- **Lifetime** - 1 year (GitLab does not specify a refresh token lifetime; this value is recommended by Aembit.)

12. Click **Save** to save your changes on the Credential Provider.

13. In Aembit UI, click the **Authorize** button. You are directed to a page where you can review the access request. Click **Authorize** to complete the OAuth 2.0 Authorization Code flow. You should see a success page and be redirected to Aembit automatically. You can also verify your flow is complete by checking the **State** value in the Credential Provider. After completion, it should be in a **Ready** state.



Credential Providers are systems that provide various types of credentials, including OAuth tokens, API keys, or username and password pairs.

In scenarios where third-party token services implement the OAuth 2.0 Client Credentials flow or Workload Identity Federation, Aembit can automatically request "access tokens" on behalf of Client Workloads.

For simple credential types like API keys or passwords, Aembit can natively store and retrieve these items when needed without relying on a third party.

Name: Example DevOps Credential Provider

Description (Optional): This is an example DevOps Credential Provider

CREDENTIAL TYPE: OAuth 2.0 Authorization Code

Client Id: bc0822635f031e1cd11b558c113d04dae2439d10843c97e35e0767f1e80e5ceb

Client Secret: ••••••••

Scopes: read_user

Base Authorization URL: https://gitlab.com/oauth

Callback URL: https://68ad91.aembit-eng.com/api/v1/credential-providers/8bcc5196-e86d-4440-9e38-

✓ PKCE Required

Lifetime: 6 Months

State: Ready

ADDITIONAL PARAMETERS

+ New Parameter

TAGS

+ New Parameter

Caution

Once the set lifetime ends, the retrieved credential will expire and no longer be active. Aembit will notify you before this happens. Please ensure you reauthorize your credential before it expires.

Client Workload Configuration

Aembit now handles the required credentials for API access to your GitLab instance, eliminating the need for you to manage them directly. You can safely remove any previously used credentials from the Client Workload.

If you access the Server Workload through an SDK or library, it is possible that the SDK/library may still require credentials to be present for initialization purposes. In this scenario, you can provide placeholder credentials. Aembit will overwrite these placeholder credentials with the appropriate ones during the access process.

Access Policy

Create an Access Policy for a Client Workload to access the GitLab REST API Server Workload. Assign the newly created Credential Provider to this Access Policy.

Required Features

You will need to configure the TLS Decrypt feature to work with the GitLab REST API Server Workload.

Now That You've Completed the Recipe…

As organizations like yours continue to expand their use of GitLab to automate and orchestrate software delivery, the question is not whether workloads should authenticate securely, but how that can be accomplished with consistency, auditability, and minimal operational friction. The guidance offered here is intended to assist teams in reducing exposure associated with static credentials and establishing practices that scale with their environments.

For teams seeking to go further and reach maturity faster, the Aembit Workload IAM Platform acts as a centralized identity and access provider for your GitLab workloads. We bridge the gap between your GitLab CI/CD jobs and the dynamic, short-lived credentials they need to access cloud resources, databases, and SaaS services.

For more information, visit aembit.io.