Les Variables et Secrets

Une bonne gestion des variables et des secrets est essentielle pour sécuriser les pipelines de déploiement, en particulier dans des environnements comme GitLab CI/CD. Ces éléments jouent un important dans la configuration des tâches d'intégration et de livraison continues, impactant directement la performance et la sécurité des applications.

Quelques Concepts et Rappels Importants

Les variables dans GitLab CI/CD peuvent être des chaînes de caractères ou des fichiers utilisés pour stocker des données pouvant varier entre les jobs ou les pipelines. Elles incluent des informations comme des chemins de fichiers, des noms d'environnement ou des configurations spécifiques. Leur flexibilité permet d'adapter les pipelines aux différents environnements, tels que la production, le développement ou les tests, sans modifier le code source.

Les secrets, quant à eux, sont des types spéciaux de variables destinés à stocker des informations sensibles. Il peut s'agir de mots de passe, de clés API, ou de certificats SSL. La principale préoccupation avec les secrets est leur sécurité. Une exposition accidentelle peut entraîner des risques de sécurité majeurs, notamment des fuites de données ou des intrusions non autorisées.

GitLab offre plusieurs moyens de déclarer ces variables et secrets. Par exemple, les variables peuvent être définies au niveau du groupe ou du projet et les secrets peuvent être masqués dans les logs de GitLab pour éviter toute exposition accidentelle.

Les Différents Types de Variables dans GitLab

GitLab CI/CD propose plusieurs types de variables, chacune adaptée à des besoins spécifiques. Comprendre ces types est essentielle pour une utilisation efficace dans vos pipelines.

Variables d’Environnement

Les variables d’environnement sont les plus courantes dans GitLab CI/CD. Elles servent à stocker des données qui peuvent être utilisées par les scripts des jobs. Par exemple, vous pouvez définir une variable `DATABASE_URL` pour stocker l’URL de votre base de données. Ces variables sont accessibles dans les scripts de pipeline en utilisant la syntaxe standard des variables d’environnement, comme `$DATABASE_URL` dans un script shell.

Variables de type Fichier

GitLab permet également de définir des variables sous forme de fichiers. Cela est utile pour les certificats, les fichiers de configuration ou tout autre type de données qui doivent être stockées sous forme de fichiers plutôt que de chaînes de texte. Ces variables sont stockées dans un fichier temporaire et le chemin d’accès au fichier est fourni en tant que variable d’environnement.

Variables Protégées

Les variables protégées sont des variables d’environnement spéciales qui ne sont accessibles que dans les branches ou les tags protégés. Elles sont utiles pour stocker des données sensibles qui ne doivent être utilisées que dans un environnement de production ou un environnement similaire sécurisé. Par exemple, vous pouvez avoir une clé API qui ne doit être utilisée que lors du déploiement en production.

Variables Masquées

Les variables masquées sont une fonctionnalité de sécurité qui empêche la valeur de la variable d’être affichée dans les logs de GitLab. C’est particulièrement important pour les secrets, comme les mots de passe ou les clés API. Lorsqu’une variable est masquée, sa valeur est remplacée par des astérisques dans les logs d’exécution de pipeline.

Exemple d’Utilisation de Variables dans un Pipeline

Voici un exemple simple montrant comment ces variables peuvent être utilisées dans un pipeline GitLab CI/CD :

```
stages:
- test
- deploy


test_job:
stage: test
script:
- echo "Running tests with DATABASE_URL=$DATABASE_URL"


deploy_job:
stage: deploy
script:
- echo "Deploying to production with PROD_API_KEY"
only:
- master
variables:
PROD_API_KEY: $PRODUCTION_API_KEY
```

Dans cet exemple, `DATABASE_URL` est une variable d'environnement standard utilisée pour les tests, tandis que `PROD_API_KEY` est une variable protégée utilisée uniquement pour les déploiements en production.

La compréhension et l'utilisation correctes de ces différents types de variables sont essentielles pour créer des pipelines CI/CD flexibles, sécurisés et efficaces dans GitLab.

Gestion des Secrets

La gestion des secrets est un aspect vital de la sécurité dans **GitLab CI/CD**. Un "secret" est une information sensible, telle qu'un mot de passe, une clé d'API, ou un certificat, qui doit être manipulée avec une extrême prudence pour éviter toute exposition ou utilisation malveillante.

La sécurisation des secrets est primordiale, car leur compromission peut entraîner des failles de sécurité majeures. Dans un pipeline CI/CD, les secrets sont souvent nécessaires pour accéder à des ressources, effectuer des déploiements, ou intégrer des services tiers.

GitLab fournit plusieurs mécanismes pour stocker de manière sécurisée les secrets. L'un d'entre eux est l'utilisation de variables protégées et masquées, comme mentionné précédemment. Il est également conseillé de limiter l'accès aux secrets aux seuls utilisateurs et processus qui en ont absolument besoin, conformément au principe du moindre privilège.

Lors de l’utilisation des secrets dans les pipelines, il est important de veiller à ce qu’ils ne soient pas exposés dans les logs ou transmis de manière non sécurisée. Voici un exemple de comment un secret peut être utilisé dans un job de pipeline :

```
deploy_job:
stage: deploy
script:
- echo "Deploying application"
- scp -i $DEPLOY_KEY package.zip user@server:/path/to/deploy
only:
- master
variables:
DEPLOY_KEY: $PRODUCTION_DEPLOY_KEY
```

Dans cet exemple, `DEPLOY_KEY` est un secret utilisé pour authentifier le processus de déploiement sur un serveur distant. Cette clé est stockée en tant que variable protégée dans GitLab et n’est exposée dans aucun log.

Définition des variables dans Gitlab

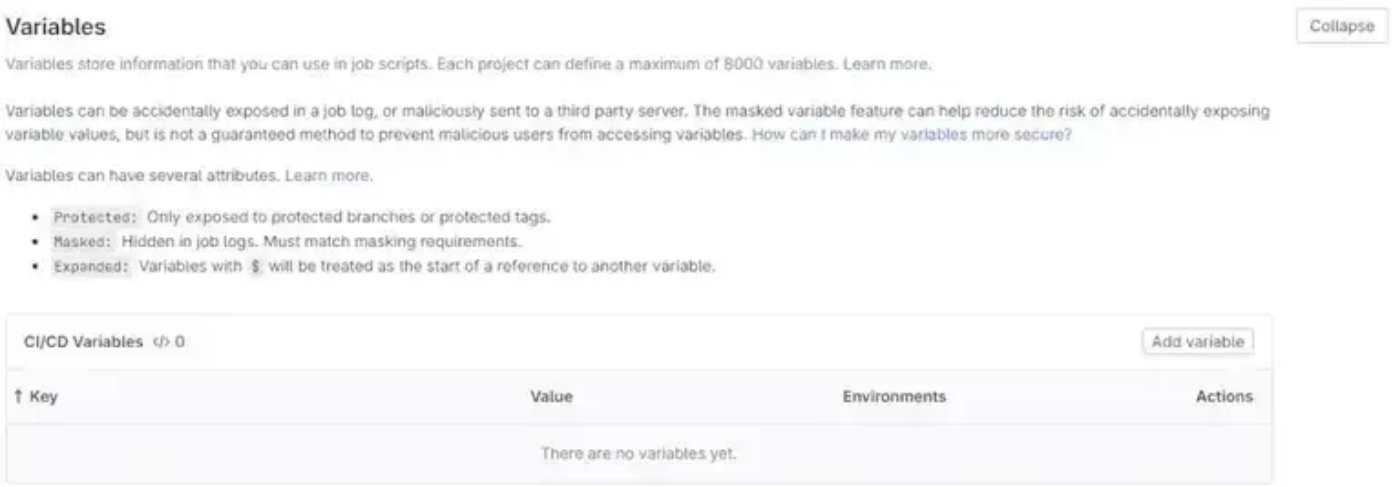
GitLab permet de définir des variables à deux niveaux principaux : au niveau du groupe et au niveau du projet. Chacun a ses avantages et utilisations spécifiques.

Définition de Variables au Niveau du Groupe

Les variables définies au niveau du groupe sont accessibles à tous les projets sous ce groupe. C’est idéal pour partager des configurations communes ou des secrets entre plusieurs projets.

Comment Définir des Variables de Groupe

1. **Accédez à la page de votre groupe** : Connectez-vous à GitLab et naviguez jusqu’à la page de votre groupe.
2. **Ouvrez les paramètres CI/CD** : Allez dans `Settings > CI/CD`.
3. **Ajoutez une variable de groupe** : Dans la section `Variables`, cliquez sur `Expand` et utilisez le formulaire pour ajouter une nouvelle variable. Vous pouvez spécifier la clé (nom de la variable), la valeur et si elle est protégée ou masquée.



Exemple d’Utilisation

Supposons que vous ayez un token d’authentification utilisé par plusieurs projets pour accéder à un service externe. En le définissant comme une variable de groupe, tous les projets sous ce groupe peuvent l’utiliser sans avoir besoin de le définir individuellement.

Définition de Variables au Niveau du Projet

Les variables de projet sont spécifiques à un projet donné. Elles sont utilisées pour stocker des configurations ou des secrets qui ne sont pertinents que pour ce projet.

Comment Définir des Variables de Projet

1. **Accédez à votre projet** : Connectez-vous à GitLab et naviguez jusqu’à la page de votre projet.
2. **Ouvrez les paramètres CI/CD** : Allez dans `Settings > CI/CD`.
3. **Ajoutez une variable de projet** : Dans la section `Variables`, cliquez sur `Expand` et ajoutez votre variable. Vous avez également les options de la protéger ou de la masquer.

Exemple d’Utilisation

Si votre projet nécessite une clé API spécifique pour se connecter à une base de données, vous pouvez la définir comme une variable de projet. Elle sera isolée de tout autre projet et ne sera disponible que pour les pipelines de ce projet spécifique.

Bonnes Pratiques

- Utilisez des variables de groupe pour des configurations communes à plusieurs projets, comme des informations d’authentification pour des outils ou services partagés.
- Privilégiez les variables de projet pour des informations spécifiques à un seul projet, afin de renforcer la sécurité et la clarté.

- Utilisez des noms de variables clairs et descriptifs pour faciliter la compréhension et la maintenance des pipelines.

Précédence des variables

La notion de “précédence des variables” fait référence à la hiérarchie selon laquelle les variables sont recherchées et utilisées lorsque plusieurs variables du même nom sont définies à différents niveaux.

Dans **GitLab CI/CD**, les variables peuvent être définies à différents niveaux, et leur priorité varie en fonction de l’endroit où elles sont déclarées. Voici les niveaux de priorité des variables dans **GitLab CI/CD**, du plus élevé au plus bas :

1. **Variables de projet :** Les variables de projet sont définies au niveau du projet GitLab. Elles ont la priorité la plus élevée et sont accessibles à tous les pipelines et jobs du projet. Vous pouvez les définir dans les paramètres du projet via l’interface Web GitLab.
2. **Variables de groupe :** Les variables de groupe sont définies au niveau du groupe GitLab qui contient le projet. Elles ont une priorité inférieure aux variables de projet. Si une variable de projet a le même nom qu’une variable de groupe, la variable de projet a la priorité.
3. **Variables de CI/CD pipeline (variables d’environnement) :** Les variables d’environnement sont définies au niveau du pipeline **GitLab CI/CD** dans le fichier `.gitlab-ci.yml`. Elles ont une priorité inférieure aux variables de projet et de groupe. Vous pouvez les définir directement dans votre fichier de configuration de pipeline et elles seront disponibles uniquement pour ce pipeline spécifique.

Voici comment la priorité des variables fonctionne en pratique :

- Si une variable est définie à la fois au niveau du projet et du groupe avec le même nom, la variable de projet a la priorité et sera utilisée dans les pipelines du projet.
- Si une variable est définie au niveau du groupe, mais pas au niveau du projet, elle sera utilisée dans les pipelines de tous les projets appartenant à ce groupe, sauf si un projet définit sa propre variable avec le même nom.
- Si une variable d’environnement est définie dans le fichier `.gitlab-ci.yml` pour un pipeline spécifique, elle a la priorité pour ce pipeline, même si une variable du même nom est définie au niveau du projet ou du groupe.

Il est important de comprendre cette hiérarchie des variables dans **Gitlab CI/CD**, car elle vous permet de gérer efficacement les secrets et les configurations spécifiques à différents niveaux de votre organisation GitLab. Vous pouvez utiliser cette hiérarchie pour contrôler l’accès aux informations sensibles, tout en permettant la personnalisation des pipelines au niveau du projet.

Bonnes Pratiques pour les Variables et Secrets

Dans ce chapitre, je mets l'accent sur les meilleures pratiques pour gérer les variables et les secrets dans **GitLab CI/CD**. L'adoption de ces pratiques assure non seulement la sécurité, mais aussi l'efficacité et la fiabilité des pipelines.

Principe du Moindre Privilège

Appliquer le principe du moindre privilège est essentiel dans la gestion des variables et des secrets. Cela signifie limiter l'accès aux informations sensibles uniquement aux utilisateurs et aux processus qui en ont strictement besoin pour accomplir une tâche spécifique. Par exemple, un secret utilisé pour le déploiement ne devrait pas être accessible lors de l'exécution des tests.

Rotation des Secrets

La rotation régulière des secrets est une pratique de sécurité essentielle. Elle implique de changer périodiquement les mots de passe, les clés API et autres informations sensibles. Cette approche réduit le risque d'attaques en cas de fuite de secret.

Gestion Centralisée des Secrets

Utiliser une solution centralisée pour gérer les secrets peut grandement améliorer la sécurité. Des outils comme HashiCorp Vault, Infisical, Passbolt, Sops… ou les services de gestion des secrets cloud (AWS Secrets Manager, Azure Key Vault, etc.) offrent des fonctionnalités avancées telles que le chiffrement, le contrôle d'accès et la rotation automatique des secrets.

Audit et Journalisation

Mettre en place un système d'audit et de journalisation pour suivre l'utilisation des variables et des secrets. Cela aide à détecter rapidement toute activité anormale ou non autorisée, ce qui est essentiel pour une réponse rapide en cas de compromission.

Environnements de Variables Séparés

Il est recommandé de séparer les variables en fonction des environnements (par exemple, développement, test, production). Cela aide à éviter les erreurs communes telles que l'utilisation accidentelle de données de production dans des environnements de test.

```
stages:
- test
- deploy


test_job:
stage: test
script:
- echo "Running tests"
variables:
DATABASE_URL: $DEV_DATABASE_URL


deploy_job:
stage: deploy
script:
- echo "Deploying to production"
only:
- master
variables:
DATABASE_URL: $PROD_DATABASE_URL
DEPLOY_KEY: $PRODUCTION_DEPLOY_KEY
```

Dans cet exemple, `DATABASE_URL` est définie séparément pour les environnements de test et de production et `DEPLOY_KEY` est une variable protégée utilisée uniquement dans l'environnement de production.

Conclusion

En suivant ces pratiques, vous pouvez non seulement améliorer la sécurité de vos pipelines CI/CD, mais aussi leur fiabilité et efficacité. La gestion des variables et des secrets ne doit pas être prise à la légère, étant donné leur impact direct sur la sécurité et la performance des processus de déploiement automatique.

Revision #1
Created 3 February 2025 17:42:03 by Nicolas
Updated 3 February 2025 18:09:18 by Nicolas