Les Environnements

Dans ce guide, je vais vous guider à travers la configuration et l’utilisation des environnements dans GitLab CI/CD, en mettant l’accent sur les meilleures pratiques et les stratégies avancées.

Comprendre les environnements Gitlab CI/CD

Un environnement dans le contexte de GitLab CI/CD et plus largement dans le développement logiciel, désigne un contexte ou un cadre dans lequel une application ou un logiciel est exécuté. Cet environnement comprend les serveurs sur lesquels l’application est déployé et s’exécute, les logiciels tiers qu’elle utilise, les variables d’environnement, la configuration réseau et d’autres composants logiciels et matériels. Le concept d’environnement est central en et dans les pratiques DevOps, où différentes phases du développement et du déploiement nécessitent des contextes distincts.

Caractéristiques Principales d’un Environnement

Isolation : Chaque environnement est isolé des autres. Cela signifie que les changements effectués dans un environnement, comme le développement, n’affectent pas directement les autres, comme la production.

Configuration Spécifique : Un environnement peut avoir sa propre configuration, incluant des versions de logiciels spécifiques, des paramètres de base de données, des variables d’environnement et des politiques de sécurité adaptées à son rôle.

Rôle Dédié : Les environnements sont généralement dédiés à des tâches spécifiques. Par exemple, un environnement de développement est utilisé pour écrire et tester le code, un environnement de test pour les tests rigoureux et un environnement de production pour l'exécution du logiciel en conditions réelles.

Exemples d'Environnements

Développement : Utilisé par les développeurs pour écrire et tester le code de manière initiale. Il est souvent configuré sur les machines locales des développeurs.

Test : Un environnement qui imite de près la production, mais est utilisé pour tester le code dans des conditions contrôlées avant qu'il ne soit déployé en production.

Staging : Un environnement de pré-production qui sert de dernière étape de test. Il est conçu pour être aussi proche que possible de l'environnement de production.

Production : L'environnement où l'application est finalement déployée et accessible aux utilisateurs finaux. Il est optimisé pour la performance, la sécurité et la stabilité.

Les environnements dans Gitlab CI/CD

Dans GitLab CI/CD, ces environnements sont définis et gérés via le fichier `.gitlab-ci.yml`, permettant aux développeurs de déployer automatiquement leur application dans différents environnements. Cela aide à assurer que les applications sont développées, testées et déployées de manière systématique, automatisée et contrôlée.

Les différents types d'environnement dans Gitlab CI/CD

Il existe deux types d'environnements :

- Les **environnements statiques** ont des noms statiques comme `staging`, `testing`, `development` ou encore `production`.
- Les **environnements dynamiques** ont des noms dynamiques utilisant des variables CI/CD

Les Environnements Statiques

Les environnements statiques sont des environnements prédéfinis dans GitLab CI/CD. Ils sont généralement constants et représentent des étapes standard du cycle de vie d'une application, comme les environnements de **développement**, **test** et **production**. Ces environnements sont définis une fois et réutilisés tout au long du processus de développement.

Il est possible de créer des environnements statiques soit :

- Dans l'interface de gitlab : Operate > Environments > New Environment
- Dans votre fichier `.gitlab-ci.yml` :

```
deploy_staging:
stage: deploy
script:
- echo "Deploy to staging server"
environment:
name: staging
url: https://staging.example.com
```

Ici, `staging` est un environnement statique. Il est défini une seule fois et reste le même quel que soit le nombre de déploiements effectués. L'avantage des environnements statiques réside dans leur simplicité et leur prévisibilité. Ils sont idéaux pour des scénarios de déploiement standardisés et ne nécessitent pas de configuration supplémentaire pour chaque déploiement.

Environnements Dynamiques

Les environnements dynamiques, en revanche, sont générés à la volée en fonction de certaines conditions ou actions, comme la création d'une nouvelle branche ou d'une merge request. Ils sont extrêmement utiles pour des scénarios tels que le test de nouvelles fonctionnalités, où vous souhaitez créer un environnement de test unique pour chaque branche de fonctionnalité sans affecter l'environnement principal de test.

Voici un exemple de configuration pour un environnement dynamique :

```
deploy_review:
stage: deploy
script: echo "Déploiement de l'environnement de revue pour $CI_COMMIT_REF_NAME"
environment:
name: review/$CI_COMMIT_REF_NAME
url: https://$CI_COMMIT_REF_NAME.example.com
only:
- branches
except:
- main
- develop
```

Dans cet exemple, un nouvel environnement de revue est créé à chaque fois qu'une nouvelle branche est poussée, à l'exception des branches `main` et `develop`. Cela signifie que chaque nouvelle branche aura son propre environnement unique, basé sur le nom de la branche. Ces environnements sont dynamiques, car ils sont créés et détruits dynamiquement en fonction du flux de travail de développement.

En résumé, les environnements statiques sont constants et prévisibles, idéaux pour les déploiements réguliers, tandis que les environnements dynamiques offrent flexibilité et spécificité, adaptés aux besoins de développement et de test en constante évolution.

Variables CI/CD propre aux environnements

Lorsque vous créez un environnement, vous indiquez son nom et son URL. Si vous souhaitez les utiliser dans vos scripts, sachez qu'ils sont accessibles par des variables spécifiques. Cela peut être particulièrement utiles pour personnaliser et automatiser des tâches en fonction de l'environnement dans lequel le pipeline CI/CD est exécuté. Elles permettent une intégration plus fine et une meilleure adaptabilité des pipelines aux différents environnements de déploiement.

Variables Propres aux Environnements GitLab

- **CI_ENVIRONMENT_NAME**
  - **Description :** Cette variable contient le nom de l'environnement pour le job en cours. Par exemple, si vous avez un environnement nommé "Production",

`CI_ENVIRONMENT_NAME` sera égal à “Production” lors de l’exécution d’un job dans cet environnement.
    - **Utilisation typique :** Souvent utilisée pour des scripts ou des configurations qui nécessitent de connaître le nom de l’environnement actuel.
- **CI_ENVIRONMENT_SLUG**
    - **Description :** Il s’agit d’une version simplifiée de `CI_ENVIRONMENT_NAME`, conçue pour être utilisée dans les URL ou les noms de domaine. Elle est automatiquement générée à partir de `CI_ENVIRONMENT_NAME` en remplaçant les caractères spéciaux.
    - **Utilisation typique :** Pratique pour créer des sous-domaines ou des chemins basés sur le nom de l’environnement, par exemple, pour des environnements de revue dynamiques.
- **CI_ENVIRONMENT_URL**
    - **Description :** Cette variable contient l’URL de l’environnement définie dans le fichier `.gitlab-ci.yml`. Cela permet d’accéder directement à l’environnement depuis l’interface utilisateur de GitLab.
    - **Utilisation typique :** Utile pour fournir un accès rapide à l’environnement depuis les rapports de pipeline, les merge requests ou les notifications.

Il est possible de surcharger la variable `$CI_ENVIRONMENT_NAME` mais la variable `$CI_ENVIRONMENT_SLUG` restera inchangé pour éviter les effets de bords.

Restreindre un environnement à une ou des branches spécifiques

Avec l’ajout de rules gitlab-ci, il est possible de restreindre des environnements à des branches définies.

Par exemple si nous souhaitons déployer sur l’environnement `review\$CI_COMMIT_REF_NAME` ci-dessus toutes les branches à l’exception de la banche master :

```
deploy_review:
stage: deploy
script:
- echo "Deploy a review app on $CI_ENVIRONMENT_SLUG"
environment:
name: review/$CI_COMMIT_REF_NAME
url: https://$CI_ENVIRONMENT_SLUG.example.com
only:
- branches
```

```
except:
- master
```

Par contre, si on ne souhaite déployer que sur l'environnement de production que la branche master :

```
deploy_prod:
stage: deploy
script:
- echo "Deploy on prod"
environment:
name: production
url: https://www.example.com
only:
- master
when: manual
```

Vous pouvez aussi ajouter la condition `when: manual` pour ne pas déclencher automatiquement ce déploiement en production.

Gestion des environnements

Arrêt d'un environnement

Il est possible de stopper un environnement, mais pour cela, il faut que dans votre CI contienne une étiquette `on_stop: nom de l'étape` et une `action: stop` associée :

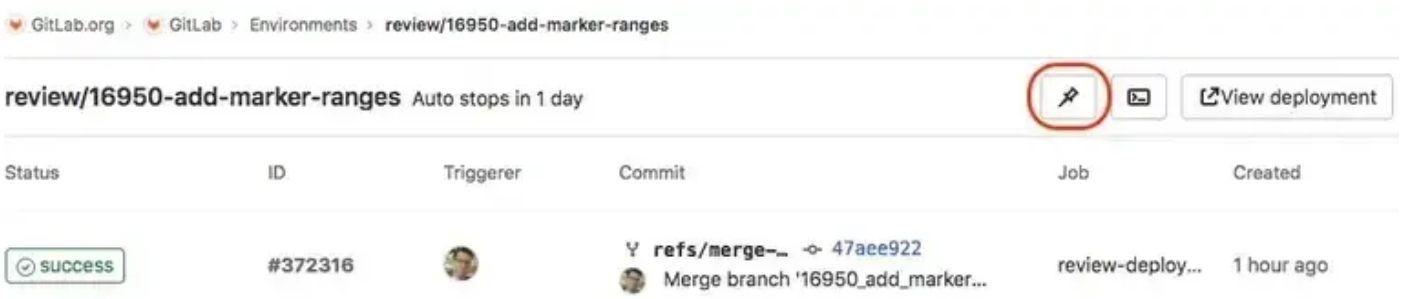
```
deploy_review:
stage: deploy
script:
- echo "Deploy a review app"
environment:
name: review/$CI_COMMIT_REF_NAME
url: https://$CI_ENVIRONMENT_SLUG.example.com
on_stop: stop_review
```

```
rules:
- if: $CI_MERGE_REQUEST_ID


stop_review:
stage: deploy
script:
- echo "Remove review app"
environment:
name: review/$CI_COMMIT_REF_NAME
action: stop
rules:
- if: $CI_MERGE_REQUEST_ID
when: manual
```

Il est également possible de stopper un environnement au bout d'un certain temps via l'ajout de l'étiquette `auto_stop_in: 1 week` :

```
review_app:
script: deploy-review-app
environment:
name: review/$CI_COMMIT_REF_NAME
on_stop: stop_review_app
auto_stop_in: 1 week
rules:
- if: $CI_MERGE_REQUEST_ID


stop_review_app:
script: stop-review-app
environment:
name: review/$CI_COMMIT_REF_NAME
action: stop
rules:
- if: $CI_MERGE_REQUEST_ID
when: manual
```

Si vous souhaitez que cette durée soit prolongée, il faut se rendre dans l'interface de gitlab : Operate > Environnements et d'épingler celui-ci.



Rollback d'un environnement

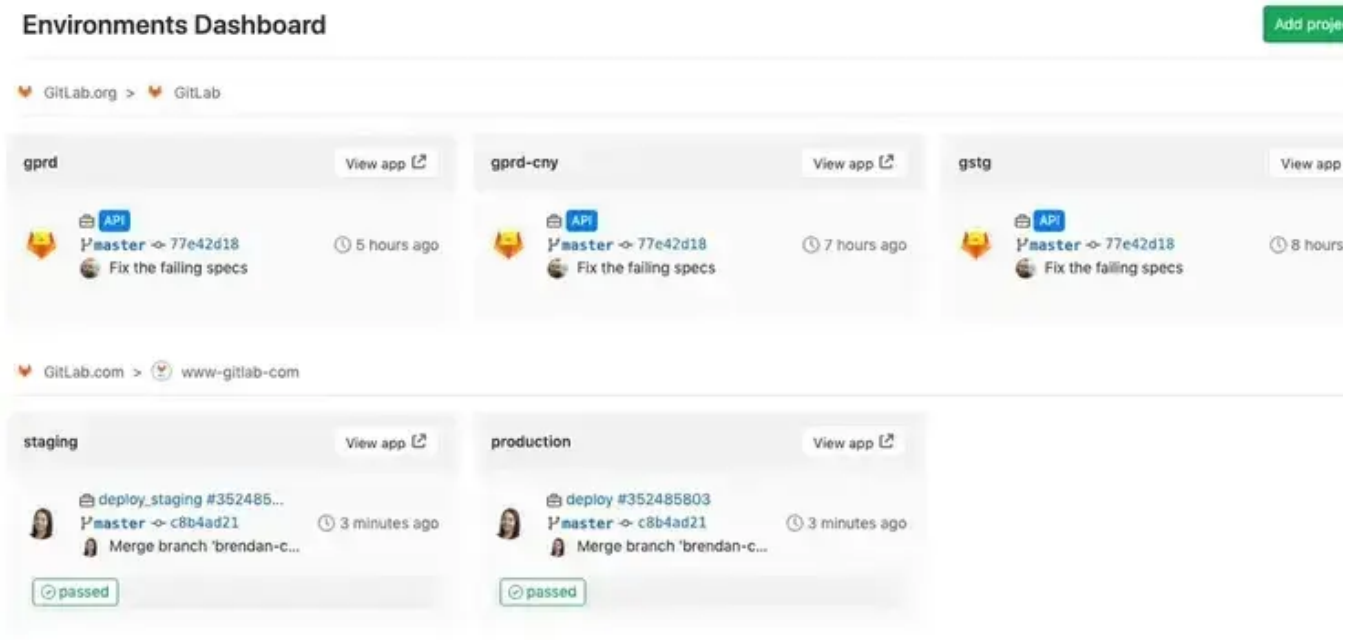
Comme dit plus haut, il est possible de faire un rollback d'environnement. Pour que cela fonctionne correctement, il faut que vos scripts le gère correctement.

loading.gif

Surveillance des Environnements

GitLab fournit des outils intégrés pour surveiller l'état et la performance des environnements. Cela inclut la surveillance de la santé des applications, le suivi des déploiements et la détection précoce des incidents.

1. **Tableaux de Bord d'Environnements :** GitLab propose des tableaux de bord spécifiques pour chaque environnement. Ces tableaux de bord affichent des informations en temps réel sur l'état des déploiements, les performances des applications et d'autres métriques clés. Ils sont essentiels pour avoir une vue d'ensemble rapide de la santé de vos environnements.



1. **Alertes et Notifications :** Configurer des alertes et des notifications est indispensable pour une réponse rapide en cas d'incidents. GitLab permet de configurer des alertes personnalisées qui peuvent être envoyées via email, Slack ou d'autres canaux de communication en cas de détection d'anomalies.

Conclusion

En parcourant les divers aspects des environnements CI/CD de GitLab, nous avons vu comment une compréhension approfondie et une utilisation efficace de ces outils peuvent transformer les processus de développement et de déploiement. De la configuration des environnements jusqu'à la surveillance et la gestion rigoureuse, GitLab CI/CD offre une plateforme robuste et flexible pour répondre aux défis complexes du DevOps moderne.

En conclusion, maîtriser les environnements GitLab CI/CD est un élément clé pour tout consultant DevOps cherchant à améliorer l'efficacité, la rapidité et la fiabilité de ses processus de déploiement. En exploitant pleinement les capacités de GitLab, vous vous assurez de répondre efficacement aux besoins en constante évolution de vos projets et de vos équipes.

Created 2025-02-03 17:47:39 UTC by Nicolas
Updated 2025-02-03 18:09:18 UTC by Nicolas