



# Documentation du pipeline CI/CD dans *CircleCI* pour une Application Web PHP

## Sommaire

1 Configuration de base du pipeline dans <code>.circleci/config.yml</code> .....	2
2 Configuration de <i>CircleCI</i> .....	2
2.1 Dans les Paramètres du projet > Variables d'environnement .....	2
2.2 Dans les Paramètres du projet > Clefs SSH .....	2
2.3 Dans les Paramètres d'organisation > Contextes .....	3
3 Ajout de jobs de metrics et lint .....	3
4 Ajout du jobs de containerisation .....	3
5 Ajout des jobs de déploiement .....	3
5.1 Serveur staging .....	3
5.2 Server production .....	4
6 Si AWS tombe .....	4

Tips : pour l'édition du fichier de configuration `.circleci/config.yml`, utiliser des extensions (comme par exemple `redhat.vscode-yaml`) qui vont garantir l'intégrité de la syntaxe `YAML`, et ainsi éviter une configuration cassée et un rejet par `CircleCI` comme sur la capture ci-dessous.



Fig. 1. – Fichier de configuration `YAML` syntaxiquement invalide, non lisible par `CircleCI`

## 1 – Configuration de base du pipeline dans `.circleci/config.yml`

Le fichier `.circleci/config.yml` est présent à la racine du projet, configuration standard pour que `CircleCI` le voit.

1. Création d'un compte `CircleCI` avec GitHub, de manière à pouvoir lier directement le dépôt à `CircleCI`
2. Définition d'un workflow `main_workflow` avec des jobs de base :
  1. `debug-info` : qui sert à afficher des variables d'environnement, le contexte d'exécution, etc.
  2. `build-setup` : job parent à tous les jobs de tests, metrics, lint, security checks, etc.
  3. `test-phpunit` : job de test pour assurer que le code est toujours test-compliant

Il faut ensuite définir une politique de contribution, à l'aide de pull-requests, contribution dans des branches comme `develop*` premièrement, puis intégration du code de releases dans des branches `release/*`, et quand on veut

Ainsi, à partir de ça, on peut définir les filtres de branches ou tag qui vont déclencher la construction d'une image docker du projet, et le déploiement sur les serveurs.

## 2 – Configuration de `CircleCI`

### 2.1 – Dans les Paramètres du projet > Variables d'environnement

Name	Value	Created at	Add Environment Variable	Import Variables
PRODUCTION_SSH_FINGERPRINT	xxxx8Sho	Feb 7, 2024, 12:30:36 AM		×
PRODUCTION_SSH_HOST	xxxx.117	Feb 7, 2024, 9:54:21 AM		×
PRODUCTION_SSH_USER	xxxxgent	Feb 6, 2024, 10:17:23 PM		×
STAGING_SSH_FINGERPRINT	xxxxclGE	Feb 5, 2024, 12:53:04 AM		×
STAGING_SSH_HOST	xxxx.126	Feb 7, 2024, 9:53:49 AM		×
STAGING_SSH_USER	xxxxgent	Feb 5, 2024, 1:09:32 AM		×

Fig. 2. – Configuration des variables d'environnement

### 2.2 – Dans les Paramètres du projet > Clefs SSH

Hostname	Fingerprint	
13.49.225.126 (Staging)	SHA256:TYibhfnCH48l/DK8VagF+CqeEZ9GIS9AW67kf2ccIGe	×
13.51.146.117 (Production)	SHA256:5tbwqoCEsUhuHljwFmW/Y30La1wa6IsoX52hdk8Sho	×

Fig. 3. – Ajout des clefs SSH des utilisateurs updater\_agent sur les serveurs de staging et production

## 2.3 – Dans les Paramètres d'organisation > Contextes

Name	Value	Created	Last Updated	
CIRCLECI_API_TOKEN	****2a1e	Feb 6, 2024 00:10 UTC	Feb 6, 2024 00:10 UTC	×
GHCR_PAT	****PoLf	Jan 28, 2024 00:24 UTC	Jan 28, 2024 00:24 UTC	×
GHCR_USERNAME	****INNE	Jan 27, 2024 20:39 UTC	Jan 27, 2024 20:39 UTC	×
INFISICAL_API_TOKEN	****ed9f	Jan 27, 2024 13:41 UTC	Jan 27, 2024 13:41 UTC	×

Fig. 4. – Définition des tokens dans les Contextes

Injection du contexte dans un job avec la directive context.

À noter que tous les tokens sauf celui d'Infisical peuvent être déplacés dans Infisical, et injectés en ligne de commande dans le pipeline avec la CLI d'Infisical.

## 3 – Ajout de jobs de metrics et lint

Ajouter de tels jobs pour avoir une meilleure visibilité sur le code, notamment au travers des rapports, trouvable dans la section *Artifacts* de chaque job.

## 4 – Ajout du jobs de containerisation

Le job qui construit l'image suit les étapes suivantes :

1. Reconstruction du nom du projet avec les variables d'environnement
2. Construction du tag avec le nom de branche et la date, ou le tag git
3. Connexion au stockage d'artifacts *GitHub Container Registry*
  - Utilisation des credentials stockés dans le contexte CircleCI des tokens
4. Construction de l'image docker à partir du code source du dépôt GitHub récupéré et injection des variables d'environnement adéquates
5. Tag de l'image
6. Stockage de l'image sur *GHCR*

## 5 – Ajout des jobs de déploiement

### 5.1 – Serveur staging

Avec le serveur configuré.

Connexion ssh avec injection de l'ensemble des commandes ci-dessous

1. Actualisation « sans-échec »/"multirisque" du dépôt en local

- Changements locaux ignorés
  - Modifications divergentes ignorées en local : priorité au remote/origin
2. Mise à jour des dépendances avec `composer`
  3. Overwrite du contenu du dossier du service `/var/www/html`
  4. Rechargement de FPM-PHP

## 5.2 — Server production

Connexion ssh avec injection de l'ensemble des commandes ci-dessous

1. Reconstruction du nom de l'image
2. Arrêt de suppression du container en cours d'exécution
3. Récupération de l'image distante en dernière version
4. Instanciation du container

## 6 — Si AWS tombe

1. On a restart et reconfiguré les machines
2. Récupération des adresses IPv4 publiques
3. Aller dans CircleCI > Projects > php-devops-tp > Project settings > SSH keys
  - Redéfinir (réutiliser) les clefs privées des deux serveurs, en remplaçant les hostnames par les IP copiées
4. Aller dans CircleCI > Projects > php-devops-tp > Project settings > Environment Variables
  - Mettre à jour de la même façon les IP pour les clefs `STAGING_SSH_HOST` et `PRODUCTION_SSH_HOST`