Publier des images Docker sur GitHub Container Registry (GHCR)

Guide complet pour publier et gérer des images Docker sur GitHub Container Registry (GHCR) : authentification, taggage, push et pull d'images

Systèmes & Réseaux 25 min de lecture Niveau Intermédiaire

Document généré le 25/05/2026 à 20h12 · nouvy.fr/wiki/publier-images-docker-ghcr

Sommaire

56 section(s) · 25 min de lecture

GitHub Container Registry (GHCR) est un service de registre de conteneurs Docker intégré à GitHub. Il permet de stocker, gérer et distribuer vos images Docker de manière sécurisée et gratuite pour les projets open source.

Introduction

Qu'est-ce que GHCR ?

GitHub Container Registry (ghcr.io) est le registre de conteneurs natif de GitHub, lancé en 2020. Il offre :

- **Intégration native** avec GitHub
- **Gratuit pour les projets publics**
- **Sécurité renforcée** avec les permissions GitHub
- **Workflow CI/CD** simplifié
- **Gestion des versions** via les tags
- **Compatibilité** avec Docker et Kubernetes

Pourquoi utiliser GHCR ?

- **Alternative à Docker Hub** : Plus de limites de pull pour les projets publics
- **Sécurité** : Permissions granulaires via GitHub
- **Intégration** : Fonctionne directement avec GitHub Actions
- **Gratuit** : Pour les projets open source publics

---

1 Se connecter à GHCR

Prérequis

Avant de pouvoir publier des images sur GHCR, vous devez :

1. Avoir un compte GitHub
2. Créer un **Personal Access Token (PAT)** avec les permissions appropriées

Créer un Personal Access Token

1. Allez sur **GitHub** → **Settings** → **Developer settings**
2. Cliquez sur **Personal access tokens** → **Tokens (classic)**
3. Cliquez sur **Generate new token (classic)**
4. Donnez un nom à votre token (ex: `GHCR_Docker_Token`)
5. Sélectionnez les scopes suivants :
    - `write:packages` : Pour publier des images
    - `read:packages` : Pour télécharger des images
    - `delete:packages` : (Optionnel) Pour supprimer des images

6. Cliquez sur **Generate token**
7. ⚠ **IMPORTANT** : Copiez le token immédiatement, il ne sera plus visible après !

Authentification avec Docker

Une fois votre token créé, connectez-vous à GHCR avec Docker :

```
echo $GITHUB_TOKEN | docker login ghcr.io -u <GITHUB_USERNAME> --password-stdin
```

Copier

Paramètres :

- `<GITHUB_USERNAME>` : Votre nom d'utilisateur GitHub
- `$GITHUB_TOKEN` : Votre Personal Access Token

Exemple :

```
# Définir le token dans une variable d'environnement
export GITHUB_TOKEN=ghp_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

# Se connecter à GHCR
echo $GITHUB_TOKEN | docker login ghcr.io -u fabob --password-stdin
```

Copier

Résultat attendu :

```
Login Succeeded
```

Copier

Alternative : Authentification avec fichier

Si vous préférez ne pas utiliser une variable d'environnement :

```
# Créer un fichier avec votre token
echo "ghp_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" | docker login ghcr.io -u fabob --password-
stdin
```

Copier

Vérifier la connexion

Pour vérifier que vous êtes bien connecté :

```
docker login ghcr.io
```

Copier

Si vous êtes déjà authentifié, vous verrez :

```
Authenticating with existing credentials...
Login Succeeded
```

Copier

---

2 Taguer ton image Docker

Format de nommage GHCR

Docker utilise le format suivant pour GHCR :

```
ghcr.io/<USERNAME>/<REPOSITORY>:<TAG>
```

Copier

Composants :

- `ghcr.io` : Le registre GitHub Container Registry
- `<USERNAME>` : Votre nom d'utilisateur GitHub
- `<REPOSITORY>` : Le nom de votre dépôt/image
- `<TAG>` : La version/tag de l'image (ex: `latest`, `1.0`, `v1.2.3`)

Exemple de taggage

Supposons que vous avez une image locale nommée `glpi-full` :

```
docker tag glpi-full ghcr.io/fabob/glpi-full:latest
```

Copier

Explication :

- `glpi-full` : Le nom de votre image locale
- `ghcr.io/fabob/glpi-full:latest` : Le nom complet pour GHCR
  - `fabob` : Votre nom d'utilisateur GitHub
  - `glpi-full` : Le nom du repository
  - `latest` : Le tag (version)

Taguer avec différentes versions

Vous pouvez taguer la même image avec plusieurs tags :

```
# Tag latest
docker tag glpi-full ghcr.io/fabob/glpi-full:latest

# Tag avec numéro de version
docker tag glpi-full ghcr.io/fabob/glpi-full:1.0

# Tag avec version complète
docker tag glpi-full ghcr.io/fabob/glpi-full:10.0.19

# Tag avec préfixe v
docker tag glpi-full ghcr.io/fabob/glpi-full:v1.0.0
```

Copier

Vérifier les tags

Pour voir toutes vos images taguées :

```
docker images | grep ghcr.io
```

Copier

Exemple de sortie :

```
ghcr.io/fabob/glpi-full    latest     abc123def456    2 hours ago    1.2GB
ghcr.io/fabob/glpi-full    1.0        abc123def456    2 hours ago    1.2GB
ghcr.io/fabob/glpi-full    10.0.19    abc123def456    2 hours ago    1.2GB
```

Copier

Bonnes pratiques de taggage

- **Utiliser** `latest` pour la version la plus récente
- **Utiliser des numéros de version** sémantiques (1.0, 1.1, 2.0)
- **Éviter les tags trop génériques** comme `dev`, `test`
- **Taguer avec le SHA du commit** pour la traçabilité
- **Utiliser des tags descriptifs** pour les releases

---

3 Pousser l'image sur GHCR

Commande de base

Une fois votre image taguée, poussez-la sur GHCR :

```
docker push ghcr.io/<GITHUB_USERNAME>/<REPOSITORY>:<TAG>
```

Copier

Exemple complet

```
docker push ghcr.io/fabob/glpi-full:latest
```

Copier

Pousser plusieurs tags

Si vous avez tagué votre image avec plusieurs versions, poussez-les toutes :

```
docker push ghcr.io/fabob/glpi-full:latest
docker push ghcr.io/fabob/glpi-full:1.0
docker push ghcr.io/fabob/glpi-full:10.0.19
```

Copier

Suivre la progression

Pendant le push, Docker affiche la progression :

```
The push refers to repository [ghcr.io/fabob/glpi-full]
abc123def456: Pushing [==================================================>]  1.2GB/1.2GB
def456ghi789: Pushing [==================================================>]  500MB/500MB
...
latest: digest: sha256:abc123def456... size: 1234
```

Copier

Vérifier sur GitHub

Une fois le push terminé, votre image sera visible sur GitHub :

1. Allez sur votre profil GitHub
2. Cliquez sur **Packages** (à droite de votre avatar)
3. Vous verrez votre package `glpi-full`

Gérer la visibilité du package

Par défaut, les packages sont **privés**. Pour le rendre **public** :

1. Allez sur **GitHub** → **Packages** → Votre package
2. Cliquez sur **Package settings**
3. Dans la section **Danger Zone**, cliquez sur **Change visibility**
4. Sélectionnez **Public**
5. Confirmez la modification

Note : Les packages publics sont gratuits et illimités !

---

4 Tester le pull

Pull depuis une autre machine

Pour tester que votre image est bien accessible, testez le pull depuis une autre machine ou après vous être déconnecté :

```
docker pull ghcr.io/<GITHUB_USERNAME>/<REPOSITORY>:<TAG>
```

Copier

Exemple

```
docker pull ghcr.io/fabob/glpi-full:latest
```

Copier

Authentification pour les packages privés

Si votre package est **privé**, vous devrez vous authentifier avant de pouvoir le pull :

```
# Se connecter à GHCR
echo $GITHUB_TOKEN | docker login ghcr.io -u fabob --password-stdin

# Puis pull l'image
docker pull ghcr.io/fabob/glpi-full:latest
```

Copier

Exécuter l'image

Une fois l'image téléchargée, vous pouvez l'exécuter :

```
docker run -p 8080:80 ghcr.io/fabob/glpi-full:latest
```

Copier

Explication :

- `-p 8080:80` : Mappe le port 80 du conteneur sur le port 8080 de l'hôte
- L'application sera accessible sur `http://localhost:8080`

Vérifier l'image

Pour voir les détails de l'image :

```
docker images ghcr.io/fabob/glpi-full
```

Copier

Pour inspecter l'image :

```
docker inspect ghcr.io/fabob/glpi-full:latest
```

Copier

Workflow complet : Exemple pratique

Voici un exemple complet de workflow pour publier une image Docker sur GHCR :

Étape 1 : Construire l'image

```
# Construire l'image depuis un Dockerfile
docker build -t glpi-full .
```

Copier

Étape 2 : Se connecter à GHCR

```
export GITHUB_TOKEN=ghp_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
echo $GITHUB_TOKEN | docker login ghcr.io -u fabob --password-stdin
```

Copier

Étape 3 : Taguer l'image

```
docker tag glpi-full ghcr.io/fabob/glpi-full:latest
docker tag glpi-full ghcr.io/fabob/glpi-full:1.0
```

Copier

Étape 4 : Pousser l'image

```
docker push ghcr.io/fabob/glpi-full:latest
docker push ghcr.io/fabob/glpi-full:1.0
```

Copier

Étape 5 : Vérifier sur GitHub

1. Allez sur https://github.com/fabob?tab=packages
2. Vérifiez que votre package `glpi-full` est présent

Étape 6 : Tester le pull

```
# Sur une autre machine
docker pull ghcr.io/fabob/glpi-full:latest
docker run -d -p 8080:80 --name glpi ghcr.io/fabob/glpi-full:latest
```

Copier

Intégration avec GitHub Actions

Workflow CI/CD automatique

Vous pouvez automatiser la publication avec GitHub Actions. Créez `.github/workflows/docker-publish.yml` :

```
name: Build and Push Docker Image

on:
  push:
    tags:
      - 'v*'

jobs:
  build-and-push:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Login to GHCR
        uses: docker/login-action@v2
        with:
          registry: ghcr.io
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Build and push
        uses: docker/build-push-action@v4
        with:
          context: .
          push: true
          tags: |
            ghcr.io/${{ github.repository_owner }}/glpi-full:latest
            ghcr.io/${{ github.repository_owner }}/glpi-full:${{ github.ref_name }}
```

Copier

Utiliser GITHUB_TOKEN

GitHub Actions fournit automatiquement un `GITHUB_TOKEN` avec les permissions nécessaires pour publier sur GHCR.

Bonnes pratiques

Sécurité

- **Ne jamais commiter** votre token dans le code
- **Utiliser des variables d'environnement** pour les tokens
- **Utiliser GitHub Secrets** dans GitHub Actions
- **Régénérer régulièrement** vos tokens
- **Utiliser des tokens avec des scopes minimaux**

Gestion des versions

- **Utiliser le versioning sémantique** (1.0.0, 1.1.0, 2.0.0)
- **Taguer avec le SHA du commit** pour la traçabilité
- **Maintenir un tag `latest`** pour la version courante
- **Ne pas supprimer les anciennes versions** sans raison

Performance

- **Utiliser des images multi-stage** pour réduire la taille
- **Optimiser les Dockerfiles** pour réduire les couches
- **Utiliser `.dockerignore`** pour exclure les fichiers inutiles
- **Scanner les images** pour les vulnérabilités

Organisation

- **Nommer clairement** vos repositories
- **Documenter** vos images avec des README
- **Utiliser des tags descriptifs**
- **Grouper les images liées** dans des repositories GitHub

Dépannage

Erreur : "unauthorized: authentication required"

Cause : Vous n'êtes pas authentifié ou votre token est invalide.

Solution :

```
# Vérifier que vous êtes connecté
docker login ghcr.io

# Si nécessaire, se reconnecter
echo $GITHUB_TOKEN | docker login ghcr.io -u <USERNAME> --password-stdin
```

Copier

Erreur : "denied: permission denied"

Cause : Votre token n'a pas les permissions nécessaires.

Solution :

1. Vérifiez que votre token a les scopes `write:packages` et `read:packages`
2. Régénérez un nouveau token si nécessaire

Erreur : "name unknown"

Cause : Le format du nom de l'image est incorrect.

Solution :

- Vérifiez que le format est : `ghcr.io/<USERNAME>/<REPOSITORY>:<TAG>`
- Assurez-vous que le nom d'utilisateur est correct

Image trop lente à pousser

Causes possibles :

- Image trop volumineuse
- Connexion internet lente
- Trop de couches dans l'image

Solutions :

- Optimiser votre Dockerfile
- Utiliser des images de base plus petites (Alpine)
- Utiliser des builds multi-stage

Comparaison avec Docker Hub

Caractéristique	GHCR	Docker Hub
Gratuit pour projets publics	□ Illimité	⚠ Limité (100 pulls/jour)
Intégration GitHub	□ Native	□ Non
Permissions	□ Granulaires	⚠ Basiques
CI/CD	□ GitHub Actions	⚠ Via webhooks
Stockage gratuit	□ Illimité (public)	⚠ 1 image gratuite
Popularité	⚠ Moins connu	□ Très populaire

Conclusion

GitHub Container Registry (GHCR) est une excellente solution pour publier et distribuer vos images Docker, surtout si vous utilisez déjà GitHub pour votre code source.

Points clés à retenir

1. **Authentification** : Utilisez un Personal Access Token avec les scopes `write:packages` et `read:packages`
2. **Format** : `ghcr.io/<USERNAME>/<REPOSITORY>:<TAG>`
3. **Workflow** : Build → Tag → Push → Pull
4. **Sécurité** : Ne jamais commiter vos tokens
5. **CI/CD** : Automatisez avec GitHub Actions

Prochaines étapes

- Explorez l'intégration avec Kubernetes
- Configurez des workflows CI/CD automatisés
- Découvrez les fonctionnalités avancées de GHCR
- Apprenez à gérer les packages privés et publics

Ressources complémentaires

Documentation officielle

- GitHub Container Registry Documentation
- Docker Documentation
- GitHub Actions Documentation

Articles et tutoriels

- GitHub Packages Guide
- Docker Best Practices
- Container Security

Outils utiles

- Docker Desktop
- GitHub CLI
- Trivy : Scanner de vulnérabilités pour images Docker