

Guide de Configuration Portainer - Générateur DOE

Objectif

Ce guide vous accompagne dans la configuration complète de Portainer pour le déploiement automatisé du Générateur DOE.

Prérequis

- ☒ Serveur Linux (Debian/Ubuntu recommandé) accessible sur **192.168.0.8**
 - ☒ Docker et Docker Compose installés
 - ☒ Portainer CE installé et accessible sur **http://192.168.0.8:9000**
 - ☒ Accès administrateur Portainer
-

Étape 1 : Installation de Portainer (si nécessaire)

Si Portainer n'est pas encore installé :

```
# Créer le volume pour les données Portainer
docker volume create portainer_data

# Installer Portainer CE
docker run -d \
  -p 9000:9000 \
  -p 8000:8000 \
  --name portainer \
  --restart always \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v portainer_data:/data \
  portainer/portainer-ce:latest

# Vérifier l'installation
docker ps | grep portainer
```

Accéder à Portainer : **http://192.168.0.8:9000**

Lors de la première connexion :

1. Créer un compte administrateur
 2. Sélectionner **Docker** comme environnement
 3. Cliquer sur **Get Started**
-

Étape 2 : Création de la Stack

2.1 Préparer les répertoires de données

Sur le serveur de production :

```
# Créer la structure de répertoires
sudo mkdir -p /data/generateur-doe-
data/{postgres,documents/{pdf,images},logs,dataprotection-
keys,temp,backups,pgadmin}

# Définir les permissions
sudo chown -R 1000:1000 /data/generateur-doe-data/documents
sudo chown -R 1000:1000 /data/generateur-doe-data/logs
sudo chown -R 1000:1000 /data/generateur-doe-data/dataprotection-keys
sudo chown -R 1000:1000 /data/generateur-doe-data/temp

sudo chown -R 1001:1001 /data/generateur-doe-data/postgres
sudo chown -R 1001:1001 /data/generateur-doe-data/backups

sudo chmod -R 755 /data/generateur-doe-data

# Vérifier
ls -la /data/generateur-doe-data/
```

2.2 Créer la Stack dans Portainer

1. Dans Portainer, naviguez vers **Stacks** dans le menu latéral
2. Cliquez sur **+ Add stack**
3. Remplissez les champs :
 - **Name** : `generateur-doe-production`
 - **Build method** : `Web editor`
4. Copiez le contenu du fichier `docker-compose.production.yml` dans l'éditeur
5. Configurez les **Environment variables** (optionnel) :

Nom	Valeur par défaut	Description
POSTGRES_PASSWORD	GenerateurDOE2025!	Mot de passe PostgreSQL (⚠ CHANGER EN PRODUCTION)
PGADMIN_EMAIL	cedric.tirolf@multisols.com	Email pgAdmin
PGADMIN_PASSWORD	GenerateurDOE2025!	Mot de passe pgAdmin (⚠ CHANGER EN PRODUCTION)

6. ⚠ **IMPORTANT - Configuration Pull** :

- Scrollez jusqu'à **Advanced mode**
- Activez **Enable auto-update** ☒

- Configurez :
 - Polling interval : **5 minutes**
 - Re-pull image : **Always**

7. Cliquez sur **Deploy the stack**

2.3 Vérification du déploiement

```
# Vérifier que tous les containers sont démarrés
docker ps | grep generateur-doe

# Vérifier les logs de l'application
docker logs generateur-doe-app-prod -f

# Vérifier les logs PostgreSQL
docker logs generateur-doe-postgres-prod -f

# Tester l'endpoint health
curl http://192.168.0.8:5000/health
# Réponse attendue : {"status":"Healthy"}
```

Étape 3 : Configuration du Webhook

3.1 Créer le Webhook dans Portainer

1. Dans Portainer, allez dans **Stacks**
2. Cliquez sur votre stack **generateur-doe-production**
3. Dans le menu de la stack, cliquez sur **Webhooks**
4. Cliquez sur **Add webhook**
5. Remplissez :
 - **Name** : **github-auto-deploy**
 - **Webhook type** : **Redeploy service**
 - **Service** : Sélectionnez **generateur-doe** (l'application)
6. Cliquez sur **Create webhook**
7. **IMPORTANT** : Copiez l'URL du webhook générée

```
http://192.168.0.8:9000/api/webhooks/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
```

3.2 Tester le Webhook

```
# Tester manuellement le webhook
curl -X POST "http://192.168.0.8:9000/api/webhooks/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

# Vérifier que le service redémarre
docker ps | grep generateur-doe-app-prod
```

Étape 4 : Configuration GitHub Secrets

4.1 Ajouter le Webhook à GitHub

1. Allez sur le dépôt GitHub : <https://github.com/zaytar68/GenerateurDOE>
2. Naviguez vers **Settings** → **Secrets and variables** → **Actions**
3. Cliquez sur **New repository secret**
4. Ajoutez le secret :
 - **Name** : `PORTAINER_WEBHOOK_URL`
 - **Secret** : Collez l'URL du webhook copiée précédemment

```
http://192.168.0.8:9000/api/webhooks/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

5. Cliquez sur **Add secret**

4.2 Vérifier les Permissions GitHub

1. Allez dans **Settings** → **Actions** → **General**
2. Scrollez jusqu'à **Workflow permissions**
3. Sélectionnez **Read and write permissions** ☒
4. Activez **Allow GitHub Actions to create and approve pull requests** ☒
5. Cliquez sur **Save**

Étape 5 : Création de la branche `production`

Sur votre machine de développement :

```
# Aller sur la branche main
git checkout main
git pull origin main

# Créer la branche production
git checkout -b production


# Pousser la branche vers GitHub
git push -u origin production
```

5.1 Tester le déploiement automatique

```
# Faire un petit changement (ex: README)
echo "Test déploiement automatique" >> README.md
git add README.md
git commit -m "test: vérification déploiement automatique"
git push origin production
```

5.2 Suivre le déploiement

1. GitHub Actions :

- Allez sur <https://github.com/zaytar68/GenerateurDOE/actions>
- Cliquez sur le workflow  **Deploy to Production**
- Suivez l'exécution en temps réel

2. Portainer :

- Allez dans **Stacks** → [generateur-doe-production](#)
- Vérifiez que la stack se redéploie automatiquement
- Consultez les logs : **Containers** → [generateur-doe-app-prod](#) → **Logs**

3. Application :

```
# Vérifier la nouvelle version
curl http://192.168.0.8:5000/health

# Consulter les logs
docker logs generateur-doe-app-prod --tail 50
```

Étape 6 : Configuration du Monitoring (Optionnel)

6.1 Activer les notifications d'événements

1. Dans Portainer, allez dans **Settings** → **Notifications**

2. Configurez un webhook Slack/Discord/Email :

- **Type** : Webhook
- **Endpoint URL** : Votre webhook Slack/Discord
- **Events** :
 - ☒ Stack deployment
 - ☒ Stack update
 - ☒ Container stopped

6.2 Configurer les Health Checks

Les health checks sont déjà configurés dans [docker-compose.production.yml](#) :

```
healthcheck:
  test: curl -f http://localhost:5000/health || exit 1
  interval: 30s
  timeout: 10s
  retries: 3
  start_period: 120s
```

Vérifier l'état des health checks :

```
docker inspect generateur-doe-app-prod | grep -A 10 Health
```

Étape 7 : Stratégie de Rollback

7.1 Configuration de la rétention des images

Dans GitHub Actions, la rétention est configurée pour conserver **4 dernières versions** :

```
cleanup-old-images:
  uses: actions/delete-package-versions@v5
  with:
    min-versions-to-keep: 4
```

7.2 Rollback manuel dans Portainer

Méthode 1 : Via l'interface Portainer

1. Allez dans **Images**
2. Trouvez ghcr.io/zaytar68/generateurdoe
3. Notez les tags disponibles (ex: [v2.1.2](#), [v2.1.3](#))
4. Allez dans **Stacks** → [generateur-doe-production](#)
5. Cliquez sur **Editor**
6. Modifiez la ligne :

```
image: ghcr.io/zaytar68/generateurdoe:v2.1.2 # Version précédente
```

7. Cliquez sur **Update the stack**

Méthode 2 : Via ligne de commande

```
# Sur le serveur
ssh user@192.168.0.8
```

```
# Arrêter la stack
docker compose -f /path/to/docker-compose.production.yml down

# Modifier l'image manuellement
docker pull ghcr.io/zaytar68/generateurdoe:v2.1.2

# Redémarrer avec l'ancienne version
docker compose -f /path/to/docker-compose.production.yml up -d
```

Étape 8 : Tests de Validation

8.1 Tests fonctionnels

```
# 1. Health check
curl http://192.168.0.8:5000/health
# ☒ Attendu: {"status":"Healthy"}

# 2. Test de l'interface
firefox http://192.168.0.8:5000

# 3. Test de génération PDF (via l'interface)
# - Créer un chantier
# - Ajouter des fiches techniques
# - Générer un document DOE

# 4. Vérifier les logs
docker logs generateur-doe-app-prod --tail 100
# ☒ Pas d'erreurs critiques
```

8.2 Tests de performance

```
# Test de charge basique
for i in {1..10}; do
  curl -w "\n" http://192.168.0.8:5000/health &
done
wait

# Vérifier l'utilisation des ressources
docker stats generateur-doe-app-prod --no-stream
# ☒ Mémoire < 2GB, CPU < 100%
```

8.3 Tests de redéploiement

```
# Sur votre machine de développement
git checkout production
```

```
echo "Test redéploiement $(date)" >> test.txt
git add test.txt
git commit -m "test: validation pipeline de déploiement"
git push origin production

# Attendre 2-3 minutes
# Vérifier dans GitHub Actions que le workflow s'exécute
# Vérifier dans Portainer que la stack se redéploie
# Vérifier que l'application redémarre correctement
```

🔗 Checklist de Validation Finale

- ☐ Portainer installé et accessible sur <http://192.168.0.8:9000>
- ☐ Stack [generateur-doe-production](#) déployée avec succès
- ☐ Tous les containers en état [Running](#)
- ☐ Health checks passent (PostgreSQL + Application)
- ☐ Webhook Portainer créé et testé
- ☐ Secret [PORTAINER_WEBHOOK_URL](#) configuré dans GitHub
- ☐ Branche [production](#) créée et poussée
- ☐ GitHub Actions workflow réussit
- ☐ Déploiement automatique fonctionne (test avec commit)
- ☐ Application accessible sur <http://192.168.0.8:5000>
- ☐ Génération PDF fonctionne
- ☐ Logs sans erreurs critiques
- ☐ Rollback testé et validé

📞 Support et Dépannage

Problèmes courants

✗ Webhook ne fonctionne pas

```
# Vérifier que Portainer écoute sur le bon port
netstat -tuln | grep 9000

# Tester le webhook localement
curl -X POST "http://localhost:9000/api/webhooks/xxx"
```

✗ Pull GHCR échoue

```
# Vérifier l'accès à GHCR depuis le serveur
docker pull ghcr.io/zaytar68/generateurdoe:latest

# Si authentication required :
```



```
docker login ghcr.io -u zaytar68
# Mot de passe : Personal Access Token GitHub
```

✗ Containers ne démarrent pas

```
# Vérifier les permissions des volumes
ls -la /data/generateur-doe-data/

# Vérifier les logs
docker logs generateur-doe-app-prod
docker logs generateur-doe-postgres-prod

# Redémarrer la stack
docker compose -f docker-compose.production.yml restart
```

Ressources

- [Documentation Portainer](#)
- [Portainer Webhooks](#)
- [GitHub Container Registry](#)
- [Guide de déploiement complet](#)

Dernière mise à jour : 2025-09-26 **Version :** 1.0.0