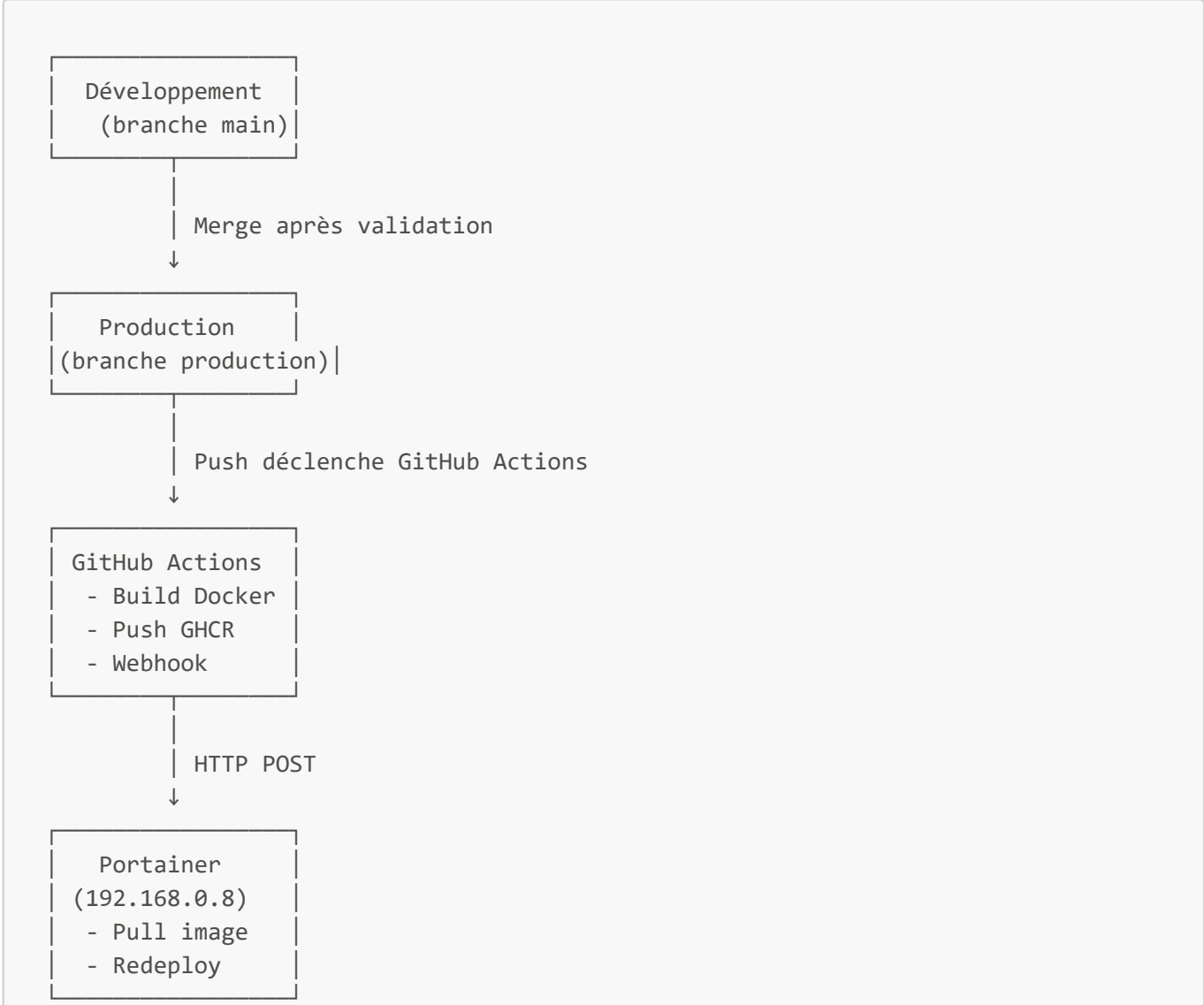
🚀 Guide de Déploiement - Générateur DOE

📋 Table des Matières

📖 Vue d'ensemble

Le système de déploiement du Générateur DOE est entièrement automatisé via GitHub Actions et Portainer.

Architecture de déploiement



🔧 Prérequis

Environnement de développement

- ☑ Git installé et configuré
- ☑ Docker (optionnel pour tests locaux)
- ☑ Accès au dépôt GitHub : `https://github.com/zaytar68/GenerateurDOE.git`

Serveur de production

- ☑ Serveur Linux avec Docker et Docker Compose installés
- ☑ Portainer CE installé et accessible sur `http://192.168.0.8:9000`
- ☑ Répertoires de données créés : `/data/generateur-doe-data/`
- ☑ Accès réseau pour pull GHCR (GitHub Container Registry)

Configuration GitHub

- ☑ Dépôt GitHub : `zaytar68/GenerateurDOE`
- ☑ Permissions sur GitHub Packages (GHCR) activées
- ☑ Secret configuré : `PORTAINER_WEBHOOK_URL`

⚙ Configuration initiale

1. Configuration GitHub Secrets

Aller dans **Settings** → **Secrets and variables** → **Actions** → **New repository secret**

Nom du Secret	Valeur	Description
`PORTAINER_WEBHOOK_URL`	`http://192.168.0.8:9000/api/webhooks/xxxxxxxx`	URL du webhook Portainer pour auto-deploy

2. Création du webhook Portainer

1. Connectez-vous à Portainer : `http://192.168.0.8:9000`
2. Naviguez vers **Stacks** → Votre stack `generateur-doe-postgresql`
3. Cliquez sur **Webhooks** dans le menu de la stack
4. Cliquez sur **Add webhook**
5. Copiez l'URL générée (format : `http://192.168.0.8:9000/api/webhooks/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`)
6. Collez cette URL dans les secrets GitHub (`PORTAINER_WEBHOOK_URL`)

3. Configuration de la stack Portainer

1. Dans Portainer, créez une nouvelle stack nommée `generateur-doe-production`
2. Utilisez le fichier docker-compose.production.yml

3. Configurez les variables d'environnement (optionnel) :

```
POSTGRES_PASSWORD=VotreMotDePasseSecurise
PGADMIN_EMAIL=votre.email@multisols.com
PGADMIN_PASSWORD=MotDePassePgAdmin
```

4. Activez **Pull latest image** dans les options de la stack
5. Déployez la stack

4. Création de la branche production

```
# Depuis la branche main
git checkout main
git pull origin main

# Créer la branche production
git checkout -b production
git push -u origin production
```

🔄 Workflow de déploiement

Déploiement standard

1. **Développement sur `main`**

```
git checkout main
# Développer, tester, commiter
git add .
git commit -m "feat: nouvelle fonctionnalité"
git push origin main
```

2. **Mise à jour de version (optionnel)**

```
# Utiliser le script de versioning
./scripts/update-version.sh 2.1.4

# Ou incrémenter automatiquement
./scripts/update-version.sh # Menu interactif
```

3. **Merge vers production**

```
git checkout production
git merge main
git push origin production
```

4. **Déploiement automatique ✨**
   - GitHub Actions s'exécute automatiquement
   - Build de l'image Docker
   - Push vers GHCR : `ghcr.io/zaytar68/generateurdoe:latest`
   - Appel du webhook Portainer
   - Portainer pull la nouvelle image et redémarre la stack

5. **Vérification**

```
# Vérifier le health check
curl http://192.168.0.8:5000/health

# Consulter les logs
docker logs generateur-doe-app-prod -f
```

Déploiement d'urgence (hotfix)

```
# Depuis production
git checkout production

# Créer une branche hotfix
git checkout -b hotfix/fix-critical-bug

# Corriger le bug
git add .
git commit -m "fix: correction bug critique"

# Merge dans production ET main
git checkout production
git merge hotfix/fix-critical-bug
git push origin production

git checkout main
git merge hotfix/fix-critical-bug
git push origin main

# Le push sur production déclenche le déploiement automatique
```

📦 Gestion des versions

Stratégie de versioning

Le projet suit **Semantic Versioning 2.0.0** : `MAJOR.MINOR.PATCH`

- **MAJOR** : Changements incompatibles de l'API
- **MINOR** : Nouvelles fonctionnalités rétrocompatibles
- **PATCH** : Corrections de bugs rétrocompatibles

Utilisation du script de versioning

```
# Menu interactif
./scripts/update-version.sh

# Incrémentation automatique
./scripts/update-version.sh 2.2.0

# Le script met à jour :
# - GenerateurDOE.csproj (Version, AssemblyVersion, FileVersion)
# - appsettings.json (ApplicationVersion)
# - docker-compose.postgresql.yml (image tag)
# - changelog.md (nouvelle entrée)
# - Tag Git (v2.2.0)
```

Synchronisation manuelle

Si vous ne voulez pas utiliser le script :

1. **Modifier** `.csproj`

   ```
   <Version>2.2.0</Version>
   <AssemblyVersion>2.2.0</AssemblyVersion>
   <FileVersion>2.2.0</FileVersion>
   ```

2. **Modifier** `appsettings.json`

   ```
   "ApplicationVersion": "2.2.0"
   ```

3. **Modifier** `changelog.md`

   ```
   ## [2.2.0] - 2025-09-26
   ### Ajouté
   - Nouvelle fonctionnalité X
   ```

4. **Créer le tag Git**

```
git tag -a v2.2.0 -m "Release version 2.2.0"
git push origin v2.2.0
```

⬅ Rollback et récupération

Rollback automatique

En cas d'échec du déploiement, Docker Swarm effectue un rollback automatique vers la version précédente (configuré dans `docker-compose.production.yml`).

Rollback manuel via Portainer

1. Connectez-vous à Portainer
2. Naviguez vers **Images**
3. Trouvez l'image `ghcr.io/zaytar68/generateurdoe`
4. Sélectionnez la version précédente (ex: `v2.1.2`)
5. Modifiez la stack pour utiliser cette version :

```
image: ghcr.io/zaytar68/generateurdoe:v2.1.2
```

6. Redéployez la stack

Rollback via ligne de commande

```
# Sur le serveur de production
cd /data/generateur-doe-data/

# Arrêter la stack
docker compose -f docker-compose.production.yml down

# Modifier l'image dans le fichier
sed -i
's|ghcr.io/zaytar68/generateurdoe:latest|ghcr.io/zaytar68/generateurdoe:v2.1.2|'
docker-compose.production.yml

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

Restauration de la base de données

```
# Sauvegarder la base actuelle (précaution)
docker exec generateur-doe-postgres-prod pg_dump -U generateur_user
GenerateurDOE_Prod > /var/backups/backup_$(date +%Y%m%d_%H%M%S).sql
```

```
# Restaurer depuis un backup
docker exec -i generateur-doe-postgres-prod psql -U generateur_user -d
GenerateurDOE_Prod < /var/backups/backup_20250925_120000.sql
```

🔧 Maintenance

Logs et monitoring

```
# Logs de l'application
docker logs generateur-doe-app-prod -f

# Logs PostgreSQL
docker logs generateur-doe-postgres-prod -f

# Logs pgAdmin
docker logs generateur-doe-pgadmin-prod -f

# Stats en temps réel
docker stats generateur-doe-app-prod
```

Backup automatique de la base de données

Créer un script de backup quotidien :

```
#!/bin/bash
# /data/generateur-doe-data/scripts/backup-db.sh

BACKUP_DIR="/data/generateur-doe-data/backups"
TIMESTAMP=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="$BACKUP_DIR/backup_$TIMESTAMP.sql"

# Créer le backup
docker exec generateur-doe-postgres-prod pg_dump -U generateur_user
GenerateurDOE_Prod > "$BACKUP_FILE"

# Compresser
gzip "$BACKUP_FILE"

# Supprimer les backups de plus de 30 jours
find "$BACKUP_DIR" -name "backup_*.sql.gz" -mtime +30 -delete

echo "✅ Backup créé: $BACKUP_FILE.gz"
```

Ajouter au crontab :

```
# Exécuter tous les jours à 2h du matin
0 2 * * * /data/generateur-doe-data/scripts/backup-db.sh
```

Nettoyage des anciennes images

```
# Sur le serveur de production
docker image prune -a --filter "until=720h" # Supprime images de plus de 30 jours

# Nettoyer uniquement les images non utilisées
docker system prune -a
```

Mise à jour de Portainer

```
# Arrêter Portainer
docker stop portainer

# Pull la dernière version
docker pull portainer/portainer-ce:latest

# Redémarrer
docker start portainer
```

Troubleshooting

Problème : Le déploiement GitHub Actions échoue

Symptômes : Workflow rouge dans GitHub Actions

Diagnostic :

```
# Consulter les logs GitHub Actions
# Aller dans Actions → Deploy to Production → Cliquer sur le run échoué
```

Solutions :

1. Vérifier que le webhook Portainer est correct dans les secrets GitHub
2. Vérifier que Portainer est accessible depuis internet (si webhook externe)
3. Vérifier les logs de build Docker dans GitHub Actions

Problème : L'image n'est pas pull par Portainer

Symptômes : Portainer utilise toujours l'ancienne version

Solutions :

```
# Sur le serveur, forcer le pull manuel
docker pull ghcr.io/zaytar68/generateurdoe:latest

# Vérifier l'image
docker images | grep generateurdoe

# Redéployer la stack dans Portainer
```

Problème : L'application ne démarre pas

Symptômes : Container en état `Restarting` ou `Exited`

Diagnostic :

```
# Consulter les logs
docker logs generateur-doe-app-prod --tail 100

# Vérifier le health check
docker inspect generateur-doe-app-prod | grep -A 10 Health
```

Solutions :

1. Vérifier la connexion à PostgreSQL

   ```
   docker exec generateur-doe-postgres-prod psql -U generateur_user -d
   GenerateurDOE_Prod -c "\conninfo"
   ```

2. Vérifier les permissions des volumes

   ```
   ls -la /data/generateur-doe-data/documents/
   # Doit appartenir à l'utilisateur 1000:1000
   ```

3. Vérifier les variables d'environnement

   ```
   docker exec generateur-doe-app-prod printenv | grep ASPNETCORE
   ```

Problème : Génération PDF échoue

Symptômes : Erreurs lors de la génération de documents PDF

Diagnostic :

```
# Vérifier Chrome/Puppeteer
docker exec generateur-doe-app-prod /usr/bin/google-chrome-stable --version

# Vérifier les logs d'erreur
docker logs generateur-doe-app-prod | grep -i puppeteer
```

Solutions :

1. Redémarrer le container

   ```
   docker restart generateur-doe-app-prod
   ```

2. Vérifier les permissions temporaires

   ```
   docker exec generateur-doe-app-prod ls -la /tmp/chrome-*
   ```

Problème : Performances dégradées

Symptômes : Application lente, timeouts

Diagnostic :

```
# Vérifier l'utilisation des ressources
docker stats generateur-doe-app-prod generateur-doe-postgres-prod

# Vérifier les connexions PostgreSQL
docker exec generateur-doe-postgres-prod psql -U generateur_user -d
GenerateurDOE_Prod -c "SELECT count(*) FROM pg_stat_activity;"
```

Solutions :

1. Augmenter les limites de ressources dans `docker-compose.production.yml`
2. Optimiser les requêtes EF Core (voir Phase 3 Performance dans CLAUDE.md)
3. Activer le cache Redis (à implémenter)

Problème : Webhook Portainer ne fonctionne pas

Symptômes : GitHub Actions réussit mais Portainer ne redémarre pas

Solutions :

1. Tester le webhook manuellement :

   ```
   ```

```
curl -X POST "http://192.168.0.8:9000/api/webhooks/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"
```

2. Vérifier que le webhook est actif dans Portainer
3. Recréer le webhook dans Portainer et mettre à jour le secret GitHub

---

Références

- GitHub Actions Documentation
- GitHub Container Registry
- Portainer Documentation
- Docker Compose Reference
- Semantic Versioning

Support

En cas de problème non résolu :

1. Consulter les logs complets (`docker logs`)
2. Vérifier le fichier CLAUDE.md pour l'architecture du projet
3. Consulter le CHANGELOG.md pour les changements récents
4. Créer une issue GitHub avec :
    - Version de l'application
    - Logs complets
    - Étapes de reproduction

Dernière mise à jour : 2025-09-26 **Version du guide** : 1.0.0 **Auteur** : Générateur DOE Team