Documentation Pipeline Jenkins - GitOps avec ArgoCD

📋 Vue d'ensemble

Cette pipeline Jenkins automatise le déploiement d'une application Angular en utilisant une approche **GitOps** avec **ArgoCD** pour la gestion des déploiements Kubernetes.

🏗️ Architecture

```
Code Angular → Jenkins → Docker Hub → GitOps Repository → ArgoCD → Kubernetes
```

🌍 Environnements

Branche Git	Environnement	Tag Docker	Branche GitOps
`main`	**prod**	`{BUILD}-prod`	`main`
`develop`	**uat**	`{BUILD}-uat`	`develop`
`feature/*`	**dev**	`{BUILD}-dev`	`feature`

🔧 Configuration Requise

Credentials Jenkins

- `dockerhubcredential` : Username/Password Docker Hub
- `gitops-credentials-argocd` : Credentials pour le repository GitOps
- `github-token` : Token GitHub pour push
- `argocd-credentials` : Credentials ArgoCD (username/password)

Outils Jenkins

- `dockerlatest` : Installation Docker
- `nodelatest` : Installation Node.js

Variables d'Environnement

- `BRANCH_NAME` : Branche Git courante
- `BUILD_NUMBER` : Numéro de build Jenkins
- `WORKSPACE` : Répertoire de travail Jenkins

🚀 Étapes de la Pipeline

1. Initialize

- Configuration des outils Docker et Node.js

- Mise à jour du PATH

2. **Checkout**

- Récupération du code source depuis Git

3. **Setup Tools**

- Installation automatique de :
  - `jq` (traitement JSON)
  - `ArgoCD CLI`
  - `kustomize` (gestion des manifests Kubernetes)

4. **Image Build**

- Construction de l'image Docker
- Nomination : `angular-app-{ENV}:{TAG}`

5. **Push to Docker Registry**

- Authentification Docker Hub
- Push de l'image avec tag approprié

6. **GitOps Update**

- Clone du repository GitOps
- Mise à jour des manifests Kubernetes via kustomize
- Commit et push des modifications

7. **Trigger ArgoCD Sync**

- Information sur la synchronisation automatique
- ArgoCD détecte automatiquement les changements

8. **Verify Deployment**

- Vérification du statut de l'application ArgoCD
- Contrôle de santé et synchronisation

📝 Fonctions Utilitaires

`imageBuild(containerName, tag)`

Construit l'image Docker avec les bonnes options

pushToImage(containerName, tag, dockerUser, dockerPassword)

Authentifie et pousse l'image vers Docker Hub

updateGitOpsManifests(containerName, tag, envName, gitUser, gitPassword)

Met à jour les manifests Kubernetes dans le repository GitOps

verifyArgoCDDeployment(envName)

Vérifie le déploiement via ArgoCD CLI

getEnvName(branchName) / **getTag(buildNumber, branchName)**

Détermine l'environnement et le tag selon la branche

📧 Notifications

Envoi automatique d'email avec :

- Statut du build
- Informations sur l'image Docker
- URL de l'application
- Nom de l'application ArgoCD

🗂️ Structure GitOps Repository

```
kubernetes-argocd-angular-javasprintboot/
├── apps/
│   └── frontend/
│       └── overlays/
│           ├── dev/
│           │   └── kustomization.yaml
│           ├── uat/
│           │   └── kustomization.yaml
│           └── prod/
│               └── kustomization.yaml
```

🔍 Monitoring et Vérification

Statuts ArgoCD

- **Healthy** : Application déployée avec succès
- **Progressing** : Déploiement en cours
- **Degraded** : Problème détecté
- **Synced** : Synchronisé avec Git

- **OutOfSync** : En attente de synchronisation

URLs des Applications

- **Production** : `https://angular-app.votre-domaine.com`
- **UAT** : `https://angular-app-uat.votre-domaine.com`
- **Development** : `https://angular-app-dev.votre-domaine.com`

⚠️ Points d'Attention

1. **Kustomize** est installé automatiquement si non présent
2. La pipeline continue même si la vérification ArgoCD échoue
3. Les credentials sont masqués dans les logs Jenkins
4. Cleanup automatique du workspace à la fin

🛠️ Dépannage

Erreurs Communes

Kustomize non trouvé

- Vérifier l'installation dans `$HOME/bin` ou `/usr/local/bin`

Échec de connexion ArgoCD

- Vérifier les credentials `argocd-credentials`
- Contrôler la connectivité vers le serveur ArgoCD

Erreur GitOps push

- Vérifier les permissions du token GitHub
- Contrôler l'existence des branches cibles

Logs Utiles

- Console Jenkins : `${BUILD_URL}/console`
- Status ArgoCD : Accessible via l'interface web ArgoCD
- Logs Kubernetes : Via `kubectl logs`

📊 Métriques

- **Temps de build** : ~5-10 minutes selon la taille de l'application
- **Temps de déploiement** : ~2-5 minutes selon les ressources Kubernetes
- **Environnements supportés** : 3 (dev, uat, prod)

🔧 Guide d'Installation et Configuration Jenkins

📦 Installation des Dépendances (Tools)

1. Installation Docker

Étape 1 : Accéder à la configuration globale

> Dashboard Jenkins → Manage Jenkins → Global Tool Configuration

Étape 2 : Configurer Docker

1. Scrollez jusqu'à la section **"Docker"**
2. Cliquez sur **"Add Docker"**
3. Configurez :
   - **Name** : `dockerlatest` (nom utilisé dans la pipeline)
   - **Install automatically** : ✅ Coché
   - **Add Installer → "Download from docker.com"**
   - **Docker version** : `latest` ou version spécifique (ex: `20.10.21`)

Étape 3 : Sauvegarder

- Cliquez sur **"Save"** en bas de page

2. Installation Node.js

Suivre le même principe :

1. Dans **Global Tool Configuration**
2. Section **"NodeJS"**
   - Si pas visible, installer le plugin : `Manage Jenkins → Manage Plugins → Available → "NodeJS Plugin"`
3. **Add NodeJS**
   - **Name** : `nodelatest`
   - **Install automatically** : ✅
   - **Version** : Choisir version LTS (ex: `18.17.0`)
   - **Global npm packages** : (optionnel)

3. Installation d'autres outils

Pour Git :

Global Tool Configuration → Git → Add Git
Name: git-latest
Path to Git executable: /usr/bin/git (ou auto-détection)

Pour Maven (si nécessaire) :

Global Tool Configuration → Maven → Add Maven
Name: maven-latest
Install automatically: ✅
Version: 3.9.4

🔐 Configuration des Secrets (Credentials)

1. Accéder à la gestion des credentials

Chemin complet :

Dashboard Jenkins → Manage Jenkins → Manage Credentials → System → Global credentials (unrestricted)

2. Docker Hub Credentials

Étapes :

1. Cliquez sur **"Add Credentials"**
2. Configurez :
   - **Kind** : `Username with password`
   - **Scope** : `Global`
   - **Username** : Votre username Docker Hub
   - **Password** : Votre password ou token Docker Hub
   - **ID** : `dockerhubcredential` (exactement ce nom)
   - **Description** : `Docker Hub Credentials`
3. Cliquez **"OK"**

3. GitOps Repository Credentials

Kind: Username with password
Username: Votre username Git
Password: Votre token GitHub/GitLab
ID: gitops-credentials-argocd
Description: GitOps Repository Access

4. GitHub Token

Kind: Username with password
Username: Votre username GitHub
Password: Votre Personal Access Token GitHub
ID: github-token
Description: GitHub Token for GitOps

Pour créer un GitHub Token :

1. GitHub → Settings → Developer settings → Personal access tokens → Tokens (classic)
2. **Generate new token**
3. Permissions requises :
   - `repo` (Full control of private repositories)
   - `workflow` (Update GitHub Action workflows)
   - `write:packages` (si registry GitHub)

5. ArgoCD Credentials

Kind: Username with password
Username: admin (ou votre user ArgoCD)
Password: Mot de passe ArgoCD
ID: argocd-credentials
Description: ArgoCD Login Credentials

🛠️ Configuration Système Jenkins

1. Plugins Requis

Accéder aux plugins :

Manage Jenkins → Manage Plugins → Available

Plugins essentiels à installer :

- `Docker Pipeline`
- `NodeJS Plugin`
- `Git Plugin`
- `Credentials Plugin`
- `Pipeline Plugin`
- `Blue Ocean` (interface moderne, optionnel)

2. Configuration de Sécurité

Script Approval (si nécessaire) :

```
Manage Jenkins → In-process Script Approval
```

- Approuver les scripts groovy utilisés dans la pipeline

3. Configuration Email

Pour les notifications :

```
Manage Jenkins → Configure System → E-mail Notification
```

Configurez :

- **SMTP Server** : `smtp.gmail.com` (ou votre serveur)
- **Default user e-mail suffix** : `@votre-domaine.com`
- **SMTP Authentication** : ✅ avec credentials

🐳 Configuration Docker (Serveur Jenkins)

Si Docker n'est pas installé sur le serveur Jenkins :

```bash
# Sur le serveur Jenkins (Ubuntu/Debian)
sudo apt-get update
sudo apt-get install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker

# Ajouter l'utilisateur jenkins au groupe docker
sudo usermod -aG docker jenkins
sudo systemctl restart jenkins
```

🔍 Vérification de la Configuration

Test des outils :

1. Créer un job de test
2. Dans le script pipeline, ajouter :

```groovy
```

```
pipeline {
    agent any
    stages {
        stage('Test Tools') {
            steps {
                script {
                    def dockerHome = tool 'dockerlatest'
                    def nodeHome = tool 'nodelatest'
                    sh "${dockerHome}/bin/docker --version"
                    sh "${nodeHome}/bin/node --version"
                }
            }
        }
    }
}
```

Test des credentials :

```
groovy

withCredentials([usernamePassword(credentialsId: 'dockerhubcredential', usernameVariable: 'USER', passwordVariable: '
    echo "Credentials loaded successfully for user: ${USER}"
}
```

📋 Checklist Configuration

- [ ] Docker tool configuré (`dockerlatest`)
- [ ] Node.js tool configuré (`nodelatest`)
- [ ] Credential Docker Hub (`dockerhubcredential`)
- [ ] Credential GitOps (`gitops-credentials-argocd`)
- [ ] Credential GitHub Token (`github-token`)
- [ ] Credential ArgoCD (`argocd-credentials`)
- [ ] Plugins Docker Pipeline installé
- [ ] Plugin NodeJS installé
- [ ] Configuration email (optionnelle)
- [ ] Docker installé sur le serveur Jenkins
- [ ] Utilisateur jenkins dans le groupe docker

⚠️ Troubleshooting Configuration

Erreur "docker: command not found" :

- Vérifier l'installation Docker sur le serveur
- Vérifier que l'utilisateur jenkins est dans le groupe docker

Erreur "Permission denied" Docker :

```bash
sudo systemctl restart docker
sudo systemctl restart jenkins
```

Credential non trouvé :

- Vérifier l'ID exact dans Manage Credentials
- S'assurer que le scope est "Global"