RÉFÉRENCE RAPIDE BITBUCKET PIPELINES

Pipelines CI/CD, cache, artefacts, déploiements

Bases des pipelines

Fonctionnement

`bitbucket-pipelines.yml`	Fichier de config à la racine du dépôt
`Conteneurs Docker`	Chaque étape s'exécute dans son propre conteneur
`Déclencheurs`	Push, PR, tag, planification ou manuel
`Minutes de build`	Quota selon le niveau d'abonnement

Activer les pipelines

```
# Paramètres du dépôt → Pipelines → Activer
# Ajouter bitbucket-pipelines.yml à la racine
# Le premier push déclenche le pipeline
```

bitbucket-pipelines.yml

Config minimale

```
image: node:20
pipelines:
  default:
    - step:
        script:
          - npm install
          - npm test
```

Pipelines par branche

```
pipelines:
  branches:
    main:
      - step:
          script:
            - npm run build
            - npm run deploy
```

Pipelines tag et pull request

```
pipelines:
  tags:
    'v*':
      - step:
          script:
            - npm run release
  pull-requests:
    '**':
      - step:
          script:
            - npm test
```

Étapes

Options d'une étape

`name`	Nom affiché pour l'étape
`image`	Remplace l'image Docker globale
`script`	Liste des commandes shell à exécuter
`size`	1x (4 Go) ou 2x (8 Go) de mémoire
`max-time`	Délai d'expiration en minutes (défaut 120)
`trigger`	manual pour les étapes manuelles uniquement

Étapes parallèles

```
- parallel:
    - step:
        name: "Lint"
        script:
          - npm run lint
    - step:
        name: "Test"
        script:
          - npm test
```

Étape manuelle

```
- step:
    name: "Déployer en production"
    trigger: manual
    script:
      - ./deploy.sh prod
```

Variables

Types de variables

`Variables de dépôt`	Paramètres → Pipelines → Variables
`Variables de déploiement`	Limitées à un environnement de déploiement
`Variables sécurisées`	Chiffrées, masquées dans les journaux
`Variables de pipeline`	Définies directement dans le YAML

Utiliser des variables

```
pipelines:
  default:
    - step:
        script:
          - echo $MY_VAR
          - docker login -u $DOCKER_USER -p $DOCKER_PASS
```

Variables intégrées

`$BITBUCKET_COMMIT`	SHA complet du commit
`$BITBUCKET_BRANCH`	Nom de la branche
`$BITBUCKET_TAG`	Nom du tag (pipelines de tag)
`$BITBUCKET_BUILD_NUMBER`	Numéro de build incrémental
`$BITBUCKET_REPO_SLUG`	Slug du dépôt

Cache

Caches prédéfinis

```
- step:
    caches:
      - node        # ~/.npm
      - pip         # ~/.cache/pip
      - docker      # cache de couches Docker
    script:
      - npm install
      - npm test
```

Cache personnalisé

```
definitions:
  caches:
    gradle: ~/.gradle/caches
    mylibs: vendor/libs
pipelines:
  default:
    - step:
        caches:
          - gradle
        script:
          - ./gradlew build
```

Comportement du cache

`Durée`	Les caches expirent après 7 jours
`Portée`	Partagé entre tous les pipelines du dépôt
`Effacer`	Pipelines → Caches → Supprimer

Artefacts

Passer des fichiers entre étapes

```
- step:
    name: "Build"
    script:
      - npm run build
    artifacts:
      - dist/**
- step:
    name: "Deploy"
    script:
      - ls dist/    # artefacts disponibles
      - ./deploy.sh
```

Options des artefacts

`artifacts`	Motifs glob des fichiers à transmettre
`Téléchargement`	Disponible automatiquement dans les étapes suivantes
`Taille max`	1 Go par étape
`Rétention`	Disponible 14 jours après le build

Déploiements

Environnements de déploiement

```
- step:
    name: "Déployer en staging"
    deployment: staging
    script:
      - ./deploy.sh staging
- step:
    name: "Déployer en production"
    deployment: production
    trigger: manual
    script:
      - ./deploy.sh prod
```

Types d'environnements

`test`	Environnement de test
`staging`	Environnement de pré-production
`production`	Environnement de production, suivi dans le tableau de bord

Patterns courants

Build et push Docker

```
- step:
    services:
      - docker
    script:
      - docker build -t myapp:$BITBUCKET_COMMIT .
      - docker login -u $DOCKER_USER -p $DOCKER_PASS
      - docker push myapp:$BITBUCKET_COMMIT
```

Conteneurs de services

```
definitions:
  services:
    postgres:
      image: postgres:16
      variables:
        POSTGRES_DB: testdb
        POSTGRES_PASSWORD: secret
pipelines:
  default:
    - step:
        services:
          - postgres
        script:
          - npm test
```

Étape conditionnelle avec Pipe

```
- step:
    name: "Déployer sur S3"
    script:
      - pipe: atlassian/aws-s3-deploy:1.1.0
        variables:
          AWS_ACCESS_KEY_ID: $AWS_KEY
          AWS_SECRET_ACCESS_KEY: $AWS_SECRET
          S3_BUCKET: my-bucket
          LOCAL_PATH: dist/
```