RIFERIMENTO RAPIDO BITBUCKET PIPELINES

Pipeline CI/CD, caching, artifact, deployment

Basi delle Pipeline

Come Funziona

`bitbucket-pipelines.yml`	File di configurazione nella root del repository
`Container Docker`	Ogni step viene eseguito nel proprio container
`Trigger`	Push, PR, tag, pianificazione o manuale
`Build minutes`	Quota dipende dal piano

Abilitare le Pipeline

```
# Impostazioni Repository → Pipeline → Abilita
# Aggiungere bitbucket-pipelines.yml alla root del repo
# Il primo push attiva la pipeline
```

bitbucket-pipelines.yml

Configurazione Minimale

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

Pipeline Specifiche per Branch

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

Pipeline per Tag e Pull Request

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

Step

Opzioni Step

`name`	Nome visualizzato dello step
`image`	Override immagine Docker globale
`script`	Lista di comandi shell da eseguire
`size`	1x (4GB) o 2x (8GB) di memoria
`max-time`	Timeout in minuti (default 120)
`trigger`	manual per step solo manuali

Step Paralleli

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

Step Manuale

```
- step:
    name: "Deploy in Produzione"
    trigger: manual
    script:
      - ./deploy.sh prod
```

Variabili

Tipi di Variabili

`Variabili repository`	Impostazioni → Pipeline → Variabili
`Variabili deployment`	Scoped a un ambiente di deployment
`Variabili protette`	Cifrate, mascherate nei log
`Variabili pipeline`	Definite inline nello YAML

Usare le Variabili

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

Variabili Predefinite

`$BITBUCKET_COMMIT`	SHA completo del commit
`$BITBUCKET_BRANCH`	Nome del branch
`$BITBUCKET_TAG`	Nome del tag (pipeline tag)
`$BITBUCKET_BUILD_NUMBER`	Numero build incrementale
`$BITBUCKET_REPO_SLUG`	Slug del repository

Caching

Cache Predefinite

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

Cache Personalizzata

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

Comportamento della Cache

`Durata`	Le cache scadono dopo 7 giorni
`Scope`	Condivisa tra tutte le pipeline del repo
`Elimina`	Pipeline → Cache → Elimina

Artifact

Passare File tra Step

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

Opzioni Artifact

`artifacts`	Pattern glob per i file da passare
`Download`	Disponibili automaticamente negli step successivi
`Dimensione max`	1 GB per step
`Conservazione`	Disponibili per 14 giorni dopo la build

Deployment

Ambienti di Deployment

```
- step:
    name: "Deploy Staging"
    deployment: staging
    script:
      - ./deploy.sh staging
- step:
    name: "Deploy Produzione"
    deployment: production
    trigger: manual
    script:
      - ./deploy.sh prod
```

Tipi di Ambiente

`test`	Ambiente di test
`staging`	Ambiente pre-produzione
`production`	Ambiente live, monitorato nella dashboard

Pattern Comuni

Build e 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
```

Container di Servizio

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

Step Condizionale con Pipe

```
- step:
    name: "Deploy su 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/
```