REFERÊNCIA RÁPIDA BITBUCKET PIPELINES

Pipelines CI/CD, cache, artefatos, deploys

Básico de Pipelines

Como Funciona

`bitbucket-pipelines.yml`	Arquivo de configuração na raiz do repositório
`Docker containers`	Cada step executa em seu próprio container
`Triggers`	Push, PR, tag, agendamento ou manual
`Build minutes`	Cota depende do plano contratado

Habilitar Pipelines

```
# Repository Settings → Pipelines → Enable
# Add bitbucket-pipelines.yml to repo root
# First push triggers the pipeline
```

bitbucket-pipelines.yml

Configuração Mínima

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

Pipelines por Branch

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

Pipelines de Tag e Pull Request

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

Steps

Opções de Step

`name`	Nome de exibição do step
`image`	Sobrescrever imagem Docker global
`script`	Lista de comandos shell a executar
`size`	1x (4GB) ou 2x (8GB) de memória
`max-time`	Timeout em minutos (padrão 120)
`trigger`	manual para steps somente manuais

Steps em Paralelo

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

Step Manual

```
- step:
    name: "Deploy to Production"
    trigger: manual
    script:
      - ./deploy.sh prod
```

Variáveis

Tipos de Variáveis

`Repository variables`	Settings → Pipelines → Variables
`Deployment variables`	Com escopo para um ambiente de deploy
`Secured variables`	Criptografadas, mascaradas nos logs
`Pipeline variables`	Definidas inline no YAML

Usando Variáveis

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

Variáveis Embutidas

`$BITBUCKET_COMMIT`	SHA completo do commit
`$BITBUCKET_BRANCH`	Nome da branch
`$BITBUCKET_TAG`	Nome da tag (pipelines de tag)
`$BITBUCKET_BUILD_NUMBER`	Número de build incremental
`$BITBUCKET_REPO_SLUG`	Slug do repositório

Cache

Caches Predefinidos

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

Cache Personalizado

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

Comportamento do Cache

`Duration`	Caches expiram após 7 dias
`Scope`	Compartilhado entre todos os pipelines do repositório
`Clear`	Pipelines → Caches → Delete

Artefatos

Passar Arquivos Entre Steps

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

Opções de Artefatos

`artifacts`	Padrões glob para arquivos a passar
`Download`	Disponível em steps subsequentes automaticamente
`Max size`	1 GB por step
`Retention`	Disponível por 14 dias após o build

Deploys

Ambientes de Deploy

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

Tipos de Ambiente

`test`	Ambiente de testes
`staging`	Ambiente de pré-produção
`production`	Ambiente de produção, rastreado no dashboard

Padrões Comuns

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
```

Containers de Serviço

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

Step Condicional com Pipe

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