REFERENCIA RÁPIDA DE BITBUCKET PIPELINES

Pipelines CI/CD, caché, artefactos, despliegues

Fundamentos del pipeline

Cómo funciona

`bitbucket-pipelines.yml`	Archivo de configuración en la raíz del repo
`Docker containers`	Cada step se ejecuta en su propio contenedor
`Triggers`	Push, PR, tag, programado o manual
`Minutos de build`	Cuota según el plan contratado

Habilitar Pipelines

```
# Repository Settings → Pipelines → Enable
# Agregar bitbucket-pipelines.yml a la raíz del repo
# El primer push activa el pipeline
```

bitbucket-pipelines.yml

Configuración mínima

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

Pipelines por rama

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

Pipelines de tag y pull request

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

Steps

Opciones de step

`name`	Nombre visible del step
`image`	Sobreescribir la imagen Docker global
`script`	Lista de comandos shell a ejecutar
`size`	1x (4 GB) o 2x (8 GB) de memoria
`max-time`	Tiempo de espera en minutos (por defecto 120)
`trigger`	manual para steps solo manuales

Steps en 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
```

Variables

Tipos de variables

`Variables de repositorio`	Settings → Pipelines → Variables
`Variables de despliegue`	Con scope a un entorno de despliegue
`Variables seguras`	Cifradas, enmascaradas en los logs
`Variables de pipeline`	Definidas en línea en el YAML

Usar variables

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

Variables integradas

`$BITBUCKET_COMMIT`	SHA completo del commit
`$BITBUCKET_BRANCH`	Nombre de la rama
`$BITBUCKET_TAG`	Nombre del tag (pipelines de tag)
`$BITBUCKET_BUILD_NUMBER`	Número de build incremental
`$BITBUCKET_REPO_SLUG`	Slug del repositorio

Caché

Cachés predefinidos

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

Caché personalizado

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

Comportamiento del caché

`Duración`	Los cachés expiran después de 7 días
`Alcance`	Compartido entre todos los pipelines del repo
`Limpiar`	Pipelines → Caches → Delete

Artefactos

Pasar archivos entre steps

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

Opciones de artefactos

`artifacts`	Patrones glob de archivos a pasar
`Descarga`	Disponibles en steps posteriores automáticamente
`Tamaño máximo`	1 GB por step
`Retención`	Disponibles 14 días después del build

Despliegues

Entornos de despliegue

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

Tipos de entorno

`test`	Entorno de pruebas
`staging`	Entorno de pre-producción
`production`	Entorno live, rastreado en el dashboard

Patrones comunes

Build y push de Docker

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

Contenedores de servicio

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

Step condicional con 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/
```