BITBUCKET PIPELINES 快速参考

CI/CD 流水线、缓存、产物、部署

流水线基础

工作原理

`bitbucket-pipelines.yml`	仓库根目录的配置文件
`Docker containers`	每个步骤在独立容器中运行
`Triggers`	push、PR、tag、定时或手动触发
`Build minutes`	配额取决于套餐级别

启用流水线

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

bitbucket-pipelines.yml

最简配置

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

按分支定义流水线

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

Tag 与 Pull Request 流水线

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

步骤

步骤选项

`name`	步骤显示名称
`image`	覆盖全局 Docker 镜像
`script`	要执行的 shell 命令列表
`size`	1x（4GB）或 2x（8GB）内存
`max-time`	超时分钟数（默认 120）
`trigger`	设为 manual 表示仅手动触发

并行步骤

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

手动步骤

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

变量

变量类型

`Repository variables`	Settings → Pipelines → Variables
`Deployment variables`	作用域限于部署环境
`Secured variables`	加密，日志中脱敏显示
`Pipeline variables`	在 YAML 中内联定义

使用变量

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

内置变量

`$BITBUCKET_COMMIT`	完整 commit SHA
`$BITBUCKET_BRANCH`	分支名称
`$BITBUCKET_TAG`	Tag 名称（tag 流水线）
`$BITBUCKET_BUILD_NUMBER`	递增构建编号
`$BITBUCKET_REPO_SLUG`	仓库 slug

缓存

预定义缓存

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

自定义缓存

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

缓存行为

`Duration`	缓存 7 天后过期
`Scope`	同仓库所有流水线共享
`Clear`	Pipelines → Caches → Delete

产物

步骤间传递文件

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

产物选项

`artifacts`	要传递的文件 glob 模式
`Download`	后续步骤自动可用
`Max size`	每步骤最大 1 GB
`Retention`	构建后保留 14 天

部署

部署环境

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

环境类型

`test`	测试环境
`staging`	预生产环境
`production`	线上环境，在仪表板中追踪

常用模式

Docker 构建与推送

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

服务容器

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

使用 Pipe 的条件步骤

```
- 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/
```