

# Bitbucket Pipelines 빠른 참조

CI/CD 파이프라인, 캐싱, 아티팩트, 배포

## 파이프라인 기본

<b>동작 방식</b>	
<b>bitbucket-pipelines.yml</b>	저장소 루트의 설정 파일
<b>Docker containers</b>	각 스텝은 자체 컨테이너에서 실행
<b>Triggers</b>	푸시, PR, 태그, 스케줄, 수동
<b>Build minutes</b>	사용 가능 빌드 시간은 플랜 등급에 따름

## 파이프라인 활성화

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

## bitbucket-pipelines.yml

### 최소 설정

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

## 브랜치별 파이프라인

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

## 태그 & PR 파이프라인

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

## 스텝

### 스텝 옵션

<b>name</b>	스텝 표시 이름
<b>image</b>	전역 Docker 이미지 재정의
<b>script</b>	실행할 셸 명령 목록
<b>size</b>	1x (4GB) 또는 2x (8GB) 메모리
<b>max-time</b>	타임아웃(분, 기본값 120)
<b>trigger</b>	manual로 설정 시 수동 실행만 가능

### 병렬 스텝

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

## 수동 스텝

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

## 변수

### 변수 유형

<b>Repository variables</b>	Settings → Pipelines → Variables
<b>Deployment variables</b>	배포 환경에 범위 지정
<b>Secured variables</b>	암호화, 로그에서 마스킹
<b>Pipeline variables</b>	YAML 인라인 정의

### 변수 사용

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

## 내장 변수

<b>\$BITBUCKET_COMMIT</b>	전체 커밋 SHA
<b>\$BITBUCKET_BRANCH</b>	브랜치 이름
<b>\$BITBUCKET_TAG</b>	태그 이름 (태그 파이프라인)
<b>\$BITBUCKET_BUILD_NUMBER</b>	증가하는 빌드 번호
<b>\$BITBUCKET_REPO_SLUG</b>	저장소 슬러그

## 캐싱

### 사전 정의 캐시

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

### 커스텀 캐시

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

### 캐시 동작

<b>Duration</b>	캐시는 7일 후 만료
<b>Scope</b>	저장소의 모든 파이프라인에서 공유
<b>Clear</b>	Pipelines → Caches → Delete

## 아티팩트

### 스텝 간 파일 전달

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

### 아티팩트 옵션

<b>artifacts</b>	전달할 파일의 글로브 패턴
<b>Download</b>	이후 스텝에서 자동으로 사용 가능
<b>Max size</b>	스텝당 1GB
<b>Retention</b>	빌드 후 14일간 보관

## 배포

### 배포 환경

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

### 환경 유형

<b>test</b>	테스트 환경
<b>staging</b>	프리 프로덕션 환경
<b>production</b>	라이브 환경, 대시보드에서 추적

## 일반 패턴

### Docker 빌드 & 푸시

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

### 서비스 컨테이너

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

# Bitbucket Pipelines 빠른 참조

---

## 파이프를 이용한 조건부 스텝

```
- 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/
```