

BITBUCKET PIPELINES QUICK REFERENCE

CI/CD pipelines, caching, artifacts, deployments

Pipeline Basics

How It Works

- bitbucket-pipelines.yml** Config file in repo root
- Docker containers** Each step runs in its own container
- Triggers** Push, PR, tag, schedule, or manual
- Build minutes** Quota depends on plan tier

Enabling Pipelines

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

bitbucket-pipelines.yml

Minimal Config

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

Branch-Specific Pipelines

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

Tag & Pull Request Pipelines

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

Steps

Step Options

- name** Display name for the step
- image** Override global Docker image
- script** List of shell commands to run
- size** 1x (4GB) or 2x (8GB) memory
- max-time** Timeout in minutes (default 120)
- trigger** manual for manual-only steps

Parallel Steps

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

Manual Step

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

Variables

Variable Types

- Repository variables** Settings → Pipelines → Variables
- Deployment variables** Scoped to a deployment environment
- Secured variables** Encrypted, masked in logs
- Pipeline variables** Defined inline in YAML

Using Variables

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

Built-in Variables

- \$(BITBUCKET_COMMIT)** Full commit SHA
- \$(BITBUCKET_BRANCH)** Branch name
- \$(BITBUCKET_TAG)** Tag name (tag pipelines)
- \$(BITBUCKET_BUILD_NUMBER)** Incrementing build number
- \$(BITBUCKET_REPO_SLUG)** Repository slug

Caching

Predefined Caches

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

Custom Cache

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

Cache Behavior

- Duration** Caches expire after 7 days
- Scope** Shared across all pipelines in repo
- Clear** Pipelines → Caches → Delete

Artifacts

Passing Files Between Steps

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

Artifact Options

- artifacts** Glob patterns for files to pass
- Download** Available in subsequent steps automatically
- Max size** 1 GB per step
- Retention** Available for 14 days after build

Deployments

Deployment Environments

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

Environment Types

- test** Testing environment
- staging** Pre-production environment
- production** Live environment, tracked in dashboard

Common Patterns

Docker Build & Push

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

Service Containers

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

Conditional Step with Pipe

```
- step:
  name: "Deploy to S3"
  script:
    - pipe: atlassian/aws-s3-deploy:1.1.0
      variables:
        AWS_ACCESS_KEY_ID: $AWS_KEY
        AWS_ACCESS_SECRET_ACCESS_KEY: $AWS_SECRET
        S3_BUCKET: my-bucket
        LOCAL_PATH: dist/
```