

DevPipeline YAML Reference

Document ID: PRD-DP-010 **Last Updated:** 2024-02-15 **Owner:** DevPipeline Engineering **Classification:** Public

Overview

DevPipeline uses YAML configuration files to define CI/CD pipelines. This reference documents all available configuration options.

Configuration file: `.devpipeline.yml` (in repository root)

Basic Structure

```
# Pipeline metadata
name: my-application
version: "2.0"

# Global variables
variables:
  NODE_VERSION: "18"

# Pipeline stages
stages:
  - build
  - test
  - deploy

# Job definitions
jobs:
  build-app:
    stage: build
    script:
      - npm install
      - npm run build
```

Top-Level Keywords

name

Pipeline identifier.

```
name: my-application-pipeline
```

version

Configuration version. Current: "2.0"

```
version: "2.0"
```

stages

Ordered list of pipeline stages.

```
stages:
  - build
  - test
  - security
  - deploy-staging
  - deploy-production
```

variables

Global variables available to all jobs.

```
variables:
  ENVIRONMENT: production
  DEBUG: "false"
  # Multi-line values
  SSH_KEY: |
    -----BEGIN RSA PRIVATE KEY-----
    ...
```

default

Default settings applied to all jobs.

```
default:
  image: node:18-alpine
  timeout: 30m
  retry: 2
  tags:
    - docker
```

Job Configuration

Basic Job

```
jobs:
  my-job:
    stage: build
    script:
      - echo "Hello, World!"
```

stage

Assign job to a stage.

```
my-job:
  stage: test
```

script

Commands to execute (required).

```
my-job:
  script:
    - npm install
    - npm test
    # Multi-line command
    - |
      if [ "$CI" = "true" ]; then
        npm run build:ci
      fi
```

before_script / after_script

Commands run before/after main script.

```
my-job:
  before_script:
    - echo "Setting up..."
  script:
    - npm test
  after_script:
    - echo "Cleaning up..." # Runs even on failure
```

image

Container image for the job.

```
my-job:
  image: node:18-alpine
  # Or with options
  image:
    name: node:18-alpine
    entrypoint: [""]
```

services

Additional containers (databases, etc.).

```
my-job:
  services:
    - postgres:14
    - redis:7
    - name: elasticsearch:8
      alias: search
```

Variables

Job-Level Variables

```
my-job:
  variables:
    DATABASE_URL: postgres://localhost/test
```

Environment Variables from Secrets

```
my-job:
  variables:
    API_KEY: ${ secrets.API_KEY }
```

Predefined Variables

Variable	Description
\$CI	Always true in pipeline
\$CI_COMMIT_SHA	Full commit SHA
\$CI_COMMIT_SHORT_SHA	Short commit SHA (8 chars)
\$CI_COMMIT_BRANCH	Branch name
\$CI_COMMIT_TAG	Tag name (if tagged)
\$CI_PIPELINE_ID	Unique pipeline ID
\$CI_JOB_ID	Unique job ID
\$CI_PROJECT_NAME	Project name
\$CI_PROJECT_PATH	Full project path

Conditions

only / except

Control when jobs run.

```
my-job:
  only:
    - main
    - /^release-.*$/ # Regex
  except:
    - schedules
```

rules

Advanced conditional logic (recommended).

```
my-job:
  rules:
    # Run on main branch
```

```

- if: $CI_COMMIT_BRANCH == "main"
  when: always
# Run on tags
- if: $CI_COMMIT_TAG
  when: always
# Run on merge requests
- if: $CI_MERGE_REQUEST_ID
  when: manual
# Default: don't run
- when: never

```

changes

Run when specific files change.

```

my-job:
  rules:
    - changes:
      - src/**/*
      - package.json

```

exists

Run when files exist.

```

my-job:
  rules:
    - exists:
      - Dockerfile

```

Parallel Execution

parallel

Run multiple instances of a job.

```

test:
  parallel: 5
  script:
    - npm run test -- --shard=$CI_NODE_INDEX/$CI_NODE_TOTAL

```

parallel:matrix

Matrix builds.

```
test:
  parallel:
    matrix:
      - NODE_VERSION: ["16", "18", "20"]
        OS: ["ubuntu", "alpine"]
    image: node:$NODE_VERSION-$OS
    script:
      - npm test
```

Artifacts and Caching

artifacts

Files to save after job completion.

```
build:
  artifacts:
    paths:
      - dist/
      - build/
    exclude:
      - "**/*.map"
    expire_in: 1 week
    when: on_success # on_success, on_failure, always
```

artifacts:reports

Special report artifacts.

```
test:
  artifacts:
    reports:
      junit: test-results.xml
      coverage: coverage/cobertura.xml
      sast: sast-report.json
      dependency_scanning: dependency-report.json
```

cache

Cache files between pipeline runs.

```
build:
  cache:
    key: $CI_COMMIT_REF_SLUG
    paths:
      - node_modules/
      - .npm/
    policy: pull-push # pull, push, pull-push
```

Cache Key Strategies

```
# Fixed key
cache:
  key: my-cache

# Branch-specific
cache:
  key: $CI_COMMIT_REF_SLUG

# File-based (invalidate on change)
cache:
  key:
    files:
      - package-lock.json
```

Dependencies

needs

Define job dependencies (DAG).

```
deploy:
  needs:
    - build
    - test
  script:
    - deploy.sh
```

needs with Artifacts

```
deploy:
  needs:
    - job: build
      artifacts: true
    - job: test
      artifacts: false
```

dependencies

Control artifact downloads.

```
deploy:
  stage: deploy
  dependencies:
    - build # Only download artifacts from build
```

Environments

environment

Define deployment target.

```
deploy-staging:
  environment:
    name: staging
    url: https://staging.example.com
```

```
deploy-production:
  environment:
    name: production
    url: https://example.com
    on_stop: stop-production
```

Dynamic Environments

```
deploy-review:
  environment:
    name: review/${CI_COMMIT_REF_SLUG}
    url: https://${CI_COMMIT_REF_SLUG}.review.example.com
    auto_stop_in: 1 week
```

Control Flow

when

Value	Description
<code>on_success</code>	Run if previous jobs succeed (default)
<code>on_failure</code>	Run if any previous job fails
<code>always</code>	Always run
<code>manual</code>	Manual trigger required
<code>delayed</code>	Run after delay
<code>never</code>	Never run (use with rules)

```
cleanup:
  when: always
  script:
    - cleanup.sh
```

allow_failure

Continue pipeline on job failure.

```
optional-test:
  allow_failure: true
  script:
    - npm run experimental-tests
```

retry

Retry failed jobs.

```
flaky-test:
  retry:
    max: 2
  when:
    - runner_system_failure
    - stuck_or_timeout_failure
```

timeout

Job timeout.

```
long-test:
  timeout: 2h 30m
```

Triggers

trigger

Trigger child pipeline.

```
trigger-deploy:
  trigger:
    project: myorg/deploy-pipeline
    branch: main
    strategy: depend
```

trigger:include

Include local pipeline config.

```
trigger-child:
  trigger:
    include: child-pipeline.yml
```

Templates

extends

Inherit from template.

```
.test-template:
  image: node:18
  before_script:
    - npm install
```

```
unit-test:
  extends: .test-template
  script:
    - npm run test:unit

integration-test:
  extends: .test-template
  script:
    - npm run test:integration
```

include

Include external configurations.

```
include:
  # Local file
  - local: .devpipeline/security.yml

  # Remote template
  - remote: https://templates.novatech.com/node.yml

  # Template from another project
  - project: novatech/pipeline-templates
    file: /templates/docker-build.yml
```

Complete Example

```
name: my-application
version: "2.0"

variables:
  NODE_VERSION: "18"

stages:
  - build
  - test
  - security
  - deploy

default:
  image: node:${NODE_VERSION}-alpine
```

```

jobs:
  build:
    stage: build
    script:
      - npm ci
      - npm run build
    cache:
      key:
        files:
          - package-lock.json
      paths:
        - node_modules/
    artifacts:
      paths:
        - dist/

  unit-test:
    stage: test
    needs: [build]
    script:
      - npm run test:unit
    artifacts:
      reports:
        junit: junit.xml
        coverage: coverage/cobertura.xml

  security-scan:
    stage: security
    needs: [build]
    script:
      - devpipeline scan --sast
    artifacts:
      reports:
        sast: sast-report.json

  deploy-staging:
    stage: deploy
    needs: [unit-test, security-scan]
    environment:
      name: staging
      url: https://staging.example.com
    script:
      - devpipeline deploy staging
    rules:
      - if: $CI_COMMIT_BRANCH == "main"

```

```
deploy-production:
  stage: deploy
  needs: [deploy-staging]
  environment:
    name: production
    url: https://example.com
  script:
    - devpipeline deploy production
  rules:
    - if: $CI_COMMIT_BRANCH == "main"
      when: manual
```

Related Documents: Quick Start Guide (PRD-DP-001), CI/CD Best Practices (PRD-DP-005), Security Scanning Guide (PRD-DP-015)