

Pipeline Architecture

Globals | Includes | Before/After | Extends

Global Defaults

```
default image | services | before_script | after_script | cache
```

variables **Cannot be specified under default**

stages **Cannot be specified under default**

Job values always override global defaults.

Include

```
include:
  - remote: 'https://gitlab.com/awesome-project/raw/master/.before-script-template.yml'
  - local: '/temp-lates/after-script-template.yml'
  - template: Auto-DevOps-gitlab-ci.yml
  - project: 'my-group/my-project'
    ref: master
    file: '/temp-lates/gitlab-ci-template.yml'
extension: .yaml | .yml
```

Before and After Scripts

```
default:
  before_script:
    - global before script
job:
  before_script:
    - execute this instead of global version
  script:
    - my command
  after_script:
    - execute this after my script
```

Extends

```
.only-important:
  only:
    - master
    - stable
  tags:
    - production
.in-docker:
  tags:
    - docker
  image: alpine
rspec:
  extends:
    - .only-important
    - .in-docker
  script:
    - rake rspec
spinach:
  extends: .in-docker
  script: rake spinach
```

Jobs Management

Stages | Parameters | Environments

Stages

```
stages:
  - .pre
  - build
  - test
  - deploy
  - .post
```

.pre and .post stages are guaranteed to be the first (.pre) or last (.post) stage in a pipeline

Disabling Jobs by Hiding Them

```
.hidden_job:
  script:
    - run test
```

temporarily 'disable' a job by prepending a dot (.)

Variables

```
variables:
  ENVIRONMENT: "staging"
  DB_URL: "postgres://postgres@postgres/db"
build:
  script: mvn build
variables:
  ENVIRONMENT: "production"
```

Environment

```
review_app:
  stage: deploy
  script: make deploy-app
  environment:
    name: review
    on_stop: stop_review_app
stop_review_app:
  stage: deploy
  variables:
    GIT_STRATEGY: none
  script: make delete-app
  when: manual
  environment:
    name: review
    action: stop
deploy as review app:
```

Environment (cont)

```
> stage: deploy
script: make deploy
environment:
  name: review/$CI_COMMIT_REF_NAME
  url: https://$CI_ENVIRONMENT_SLUG.example.com/
```

Pages

```
pages:
  stage: deploy
  script:
    - mkdir .public
    - cp -r * .public
    - mv .public public
  artifacts:
    paths:
      - public
  only:
    - master
```

Pages is a special job that is used to upload static content to GitLab that can be used to serve your website

Dependencies

```
build:osx:
  stage: build
  script: make build:osx
artifacts:
  paths:
    - binaries/
build:linux:
  stage: build
  script: make build:linux
```

Dependencies (cont)

```
> artifacts:
  paths:
    - binaries/
test:osx:
  stage: test
  script: make test:osx
dependencies:
  - build:osx
test:linux:
  stage: test
  script: make test:linux
dependencies:
  - build:linux
deploy:
  stage: deploy
  script: make deploy
```

By default, all artifacts from all previous stages are passed to the current job, but you can use the dependencies parameter to define a limited list of jobs (or no jobs) to fetch artifacts from.

Needs

```
linux:build:
  stage: build
mac:build:
  stage: build
linux:rspec:
  stage: test
  needs: ["linux:build"]
linux:rubocop:
  stage: test
  needs: ["linux:build"]
```

Needs (cont)

```
> mac:rspec:
  stage: test
  needs: ["mac:build"]
mac:rubocop:
  stage: test
  needs: ["mac:build"]
production:
  stage: deploy
```

The needs: keyword enables executing jobs out-of-order, allowing you to implement a directed acyclic graph. This lets you run some jobs without waiting for other ones, disregarding stage ordering so you can have multiple stages running concurrently.

Tags

```
job:
  tags:
    - ruby
    - postgres
osx job:
  stage:
    - build
  tags:
    - osx
  script:
    - echo " Hello, $USER! "
```

tags is used to select specific Runners from the list of all Runners that are allowed to run this project. During the registration of a Runner, you can specify the Runner's tags, for example ruby, postgres, windows, osx.



By **violaken** (violaken)
cheatography.com/violaken/

Not published yet.
Last updated 31st October, 2019.
Page 2 of 4.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

Trigger

```
staging:
  stage: deploy
  trigger: my/dep loyment

staging-b ranch:
  stage: deploy
  trigger:
    project: my/dep loyment
    branch: stable
```

trigger allows you to define downstream pipeline trigger. When a job created from trigger definition is started by GitLab, a downstream pipeline gets created.

Parallel

```
test:
  parallel: 3
  script:
    - bundle
    - bundle exec rspec -
  booster --job $CI_NO DE_ IND -
  EX/ $CI_NO DE_ TOTAL
```

parallel allows you to configure how many instances of a job to run in parallel. This value has to be greater than or equal to two (2) and less than or equal to 50.

Flow Control

Rules | Retries

rules Evaluation

```
docker build:
  script: docker build -t my-
  ima ge: $SLUG .
  rules:
    - changes:
      - Dockerfile
      when: manual
    - if: '$VAR == " string
  value"'
      when: manual
    - when: on_success

docker build:
  script: docker build -t my-
  ima ge: $SLUG .
  rules:
    - if: '$VAR == " string
  value"'
      changes:
      - Dockerfile
      - docker /sc ripts/*
      when: manual
```

To conjoin if and changes clauses with an AND, use them in the same rule.

Job Retries

```
test:
  script: rspec
  retry:
    max: 2
    when:
      - runner_sy ste -
  m_f ailure
      - stuck_ or_ tim -
  eou t_f ailure

when: always | unknow n_f ailure
| script_fa ilure | api_fa ilu
re | stuck_ or_ tim eou t_f ailu
re | runner_sy ste m_f ailure |
  missin g_d epe nde ncy_fa ilu
re | runner_un sup ported
```

Interruptible

```
stages:
  - stage1
  - stage2

step-1:
  stage: stage1
  script:
    - echo "Can be cancel ed"

step-2:
  stage: stage2
  script:
    - echo "Can not be
  cancel ed"
  int err upt ible: false
```

This value will only be used if the **automatic cancellation of redundant pipelines** feature is enabled.

Protecting Manual Jobs

```
deploy_prod:
  stage: deploy
  script:
    - echo " Deploy to
  production server "
  env iro nment:
    name: production
    urhttps: //e xam -
  ple.com
  when: manual
  only:
    - master
  all ow_ fai lure: false
```

In the protected environments settings, select the environment and add the users, roles or groups that are authorized to trigger the manual job to the Allowed to Deploy list



Artifact Management

Artifacts | Docker | Cache

Artifacts

```
job:
  artifact:
    name: "$CI_JOB_NAME -
    paths:
      - binaries/
      - .config
    expire_in: 1 week
  code_quality:
    stage: test
    script: codequality /code
    artifact:
      reports:
        codequality: gl-
        coverage: '/Code coverage:
        \d+\.\d+/'
  untracked: true | false when: on_
  success | on_failure | always |
  manual
```

Docker Image

```
image:
  name: super/ sql :ex per -
  experimental
  entrypoint: [""]
```

Docker Service

```
services:
  - name: postgres:9.4
    alias: db
    entrypoint: ["docker -
  r-entrypoint.sh "]
    command: ["postgres "]
```

Cache

```
build:
  script: mvn test
  cache:
    key: build
    untracked: true
    paths:
      - binaries/
    policy: pull
  policy : pull | push | pull-push
  untracked : true | false
```

