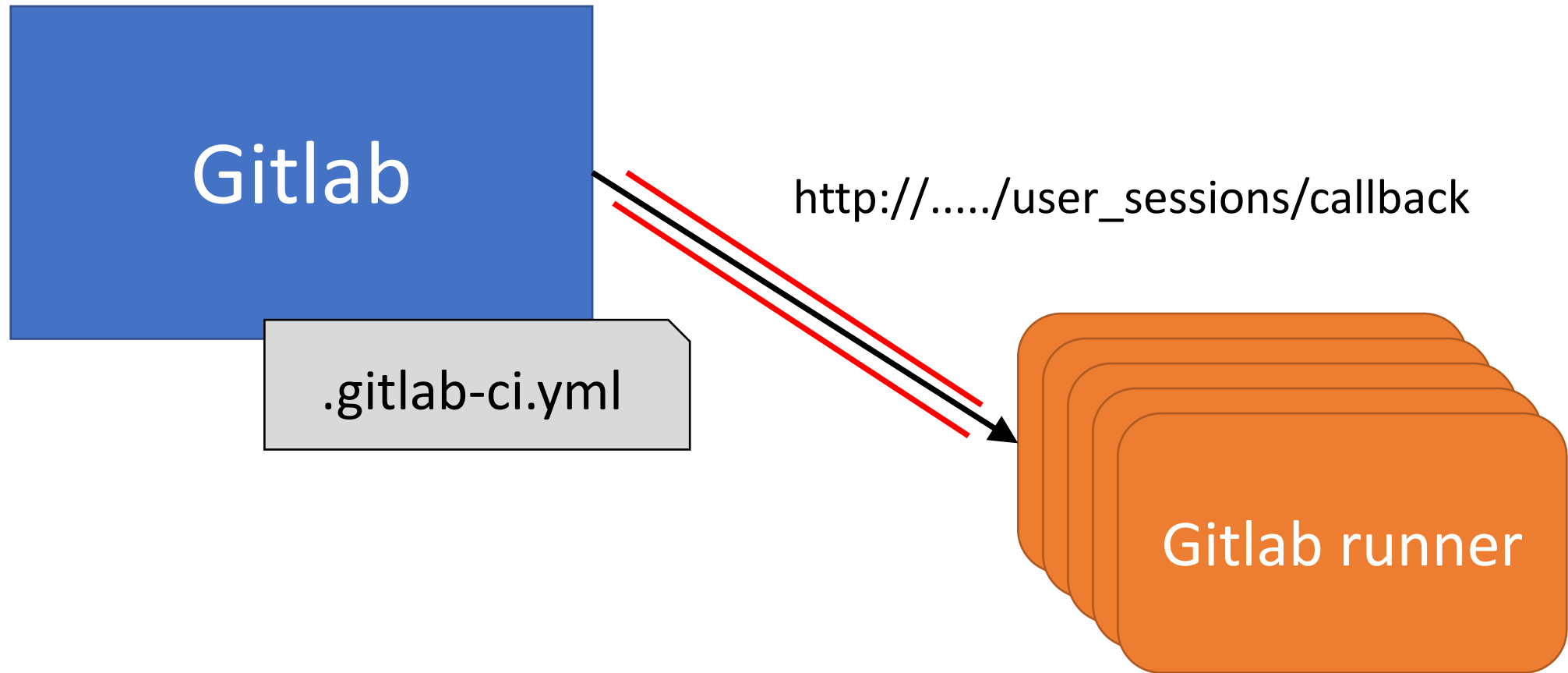


# GitLab CI

Kari Systä  
15.11.2022

# Gitlab CI

<https://docs.gitlab.com/ee/ci/>



# Types of runners

## Shared Runners

- These runners are useful for jobs multiple projects which have similar requirements. Instead of using multiple runners for many projects, you can use a single or a small number of Runners to handle multiple projects which will be easy to maintain and update.

## Specific Runners

- These runners are useful to deploy a certain project, if jobs have certain requirements or specific demand for the projects. Specific runners use *FIFO* (First In First Out) process for organizing the data with first-come first-served basis.

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

Example from:  
[https://docs.gitlab.com/ee/user/project/pages/getting\\_started/pages\\_from\\_scratch.html](https://docs.gitlab.com/ee/user/project/pages/getting_started/pages_from_scratch.html)

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```



Base Image

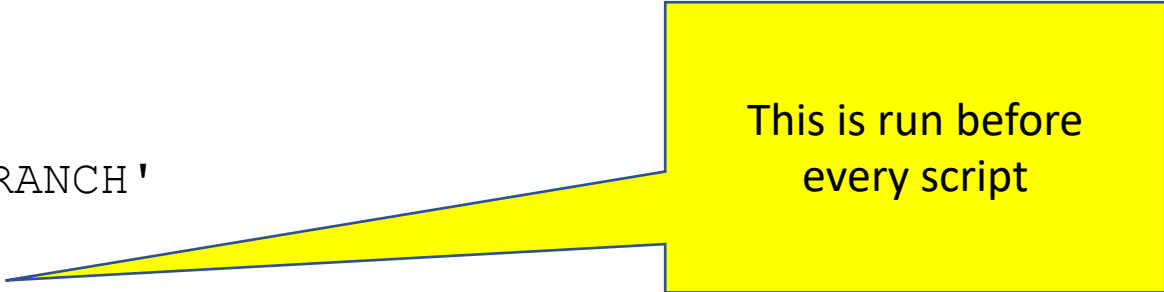
```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```



This is run before  
every script

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

Used rules

Many variables available:  
[https://docs.gitlab.com/ee/ci/variables/predefined\\_variables.html](https://docs.gitlab.com/ee/ci/variables/predefined_variables.html)

Use of rule,  
executed if rule is  
"master"

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

This is for state "deploy".

Default states are  
build, test, deploy

This is for state "test".



```
image: ruby:2.7

workflow:
  rules:
    - if: '$SCI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$SCI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$SCI_COMMIT_BRANCH != "master"'
```

Never  
mind 😊

Script to run

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```



File location

# How to install .gitlab-ci.yml?

```
git add .gitlab-ci.yml
```

```
git commit -m "Add .gitlab-ci.yml"
```

```
git push origin master
```

passed

#2913



🔗 master ↪ 43dda676

more tests



🕒 00:00:36

📅 1 month ago

passed

#2912



🔗 master ↪ 32e0f29b

more tests



🕒 00:00:36

📅 1 month ago

failed

#2911



🔗 master ↪ 8bf6c037

more tests



🕒 00:00:16

📅 1 month ago

Sphinx error:

Missing config path exercises/hello\_\_hello/config.yaml

make: \*\*\* [html] Error 1

Makefile:60: recipe for target 'html' failed

\*\*\* ERROR in compile-rst

▼

▼

ERROR: Job failed: exit code 1

variables:

TUNIPLUSSA\_ID: 'TIE23536-  
syksy2019'

GIT\_STRATEGY: none

stages:

- build
- test
- deploy

builder:

stage: build

only:

- master
- release

tags:

- plussa

artifacts:

paths:

- FULLLOG.txt

expire\_in: 2 week

script:

- tuni-rst-build

tester:

stage: test

only:

- master

tags:

- plussa

script:

- tuni-publish-to-testing

publisher:

stage: deploy

only:

- release

tags:

- plussa

script:

- tuni-publish-to-production

```
variables:  
  TUNIPLUSSA_ID: 'TIE23536-  
syksy2019'  
  GIT_STRATEGY: none
```

```
stages:  
  - build  
  - test  
  - deploy
```

```
builder:  
  stage: build  
  only:  
  - master  
  - release  
  tags:  
  - plussa  
  artifacts:  
    paths:  
    - FULLLOG.txt  
    expire_in: 2 week  
  script:  
  - tuni-rst-build
```

```
tester:  
  stage: test  
  only:  
  - master  
  tags:  
  - plussa  
  script:  
  - tuni-publish-to-testing  
  - tuni-publish-to-production
```

**Note:** The rules syntax is an improved, more powerful solution for defining when jobs should run or not. Consider using rules instead of only/except to get the most out of your pipelines.

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

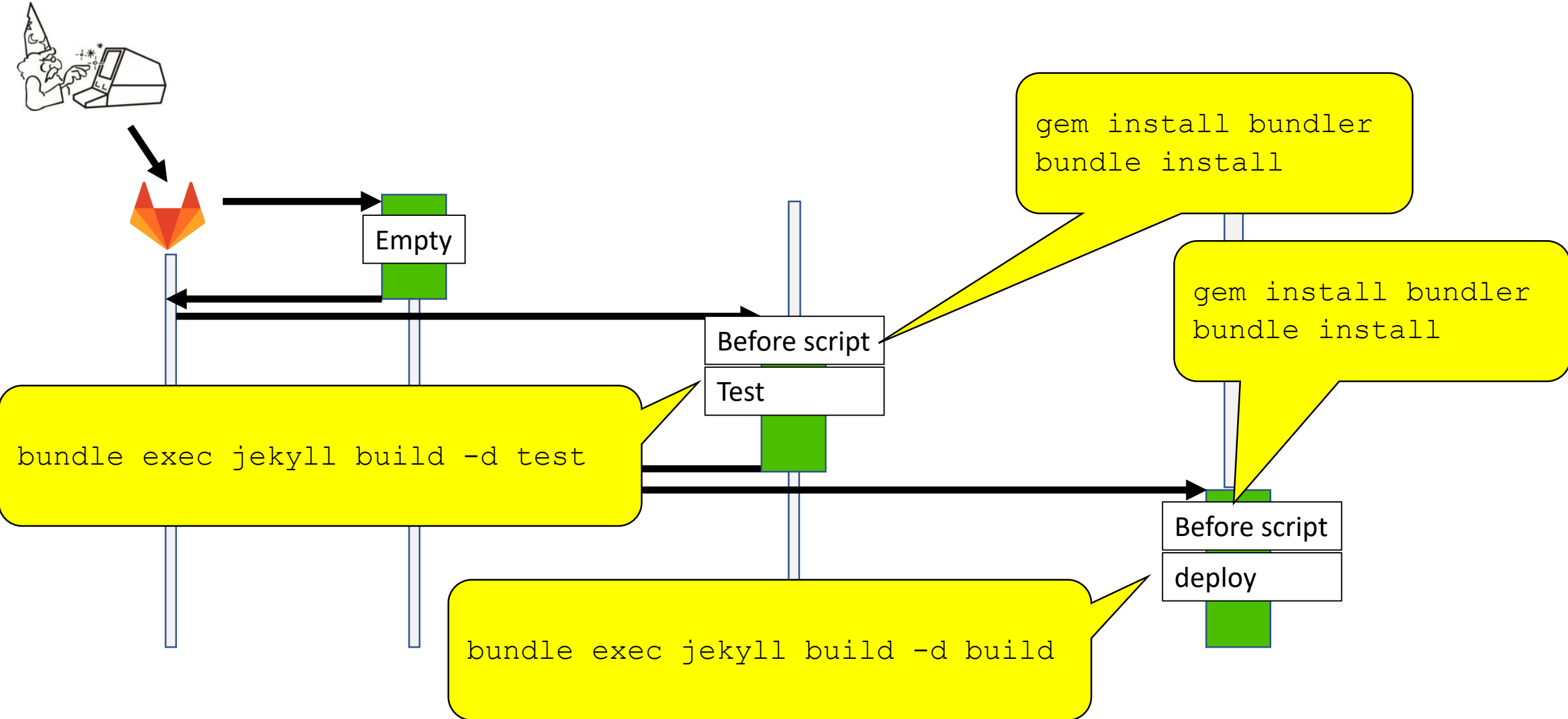
test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

Example from:  
[https://docs.gitlab.com/ee/user/project/pages/getting\\_started/pages\\_from\\_scratch.html](https://docs.gitlab.com/ee/user/project/pages/getting_started/pages_from_scratch.html)



# Is this correct?

# Why not?



# This is correct visualization!

