ATIX

The Linux & Open Source Company

# Fun with artifacts and child-pipelines in GitLab

## Include or trigger?

Jan Bundesmann

# THE Linux & Open Source Company!

Consulting

Engineering

Support

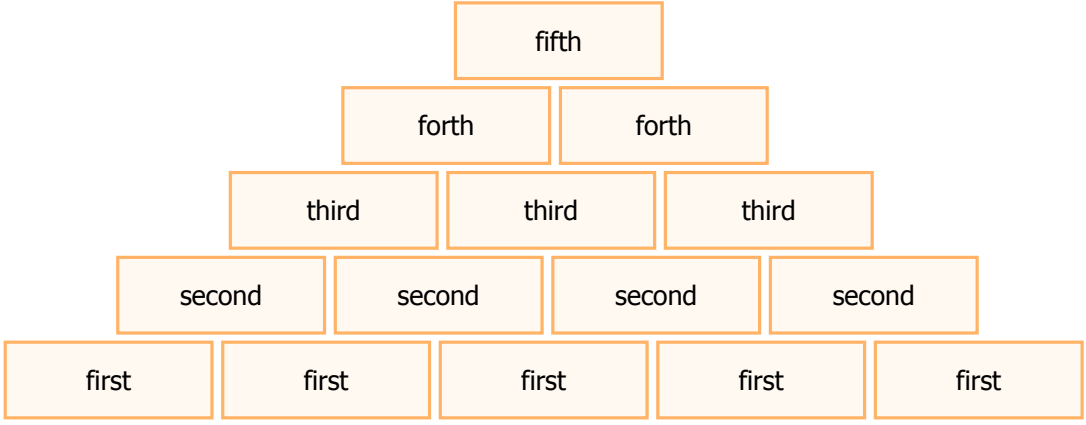
Training

AlmaLinux · amazon web services · ANSIBLE · Azure · CentOS · debian · docker · FOREMAN · git · Google Cloud Platform

Istio · kafka · KATELLO · kubernetes · KVM · ORACLE Linux · oVirt · PROXMOX · puppet · RANCHER

Red Hat · RED HAT OPENSHIFT · Rocky Linux · SALTSTACK · SUSE · Terraform · Ubuntu · vmware · Windows

# Let’s build


fifth
forth
forth
third
third
third
second
second
second
second
first
first
first
first
first

# Let’s build II

# Bricks and mortar

Why?

- Reduce **complexity**
- Keep system **flexible** and **modular**

How?

- Write one “glue” pipeline
- Pass information: **parameters** and **artifacts**
- Final artifact shall combine all intermediate steps

# Agenda

1. Includes
2. Trigger
3. Difficulties
   - Scaling Includes
   - more than just CI
   - Artifacts
4. Deciding

# Include the first repo

.gitlab-ci.yml:

```yaml
stages:
  - first

include:
  - local: variables.yaml

doit:
  stage: first
  script:
    - 'echo "first: ${information}" >> data.
        html'
  artifacts:
    paths:
      - data.html
```

variables.yaml:

```yaml
variables:
  information: This is the first repo.
```

.gitlab-ci.yml:

```yaml
include:
  - project: gent2024/first
    file: .gitlab-ci.yml
```

variables.yaml:

```yaml
variables:
  information: This is glue1-include.
```

# Quote from docs

## From https://docs.gitlab.com/ee/ci/yaml/#include

The include files are:

- Merged with those in the .gitlab-ci.yml file.
- Always evaluated first and then merged with the content of the .gitlab-ci.yml file, regardless of the position of the include keyword.

## From https://docs.gitlab.com/ee/ci/yaml/#includeproject

All nested includes are executed in the scope of the project containing the configuration file with the nested include keyword. You can use local (relative to the project containing the configuration file with the include keyword), project, remote, or template includes.

# Take home

```yaml
include:
  - project: gent2024/first
    file: .gitlab-ci.yml
  - local: variables.yaml

variables:
  information:
    value: glue1 (.gitlab-ci.yml)
```

- Included files are merged in the order they are included.
- Nested includes are evaluated relative to the included file.
- Properties are unchanged unless explicitely overwritten (e.g. stage)
- Project `first` is unchanged.

# Agenda

1. Includes
2. Trigger
3. Difficulties
   - Scaling Includes
   - more than just CI
   - Artifacts
4. Deciding

# Trigger pipeline from the first repo

.gitlab-ci.yml:

```yaml
stages:
  - first

include:
  - local: variables.yaml

doit:
  stage: first
  script:
    - 'echo "first: ${information}" >> data.
        html'
  artifacts:
    paths:
      - data.html
```

variables.yaml:

```yaml
variables:
  information: This is the first repo.
```

.gitlab-ci.yml:

```yaml
stages:
  - first
  - second
  - third

first:
  stage: first
  trigger:
    project: gent2024/first

second:
  stage: second
  trigger:
    include:
      - project: gent2024/first
        file: .gitlab-ci.yml
```

# Take home

- Trigger create downstream pipelines
- Downstream pipelines have their own stages
- Environments are created parallel to downstream pipeline
- Artifacts belong to downstream pipeline
- Variables are not included from downstream pipeline
- Parameters can be passed to downstream pipelines → also via `parallels:matrix`
- Two kinds of downstream pipelines:
  - multi-project (“trigger-project”)
  - child (“trigger-include”)

# Agenda

1 Includes

2 Trigger

3 Difficulties
- Scaling Includes
- more than just CI
- Artifacts

4 Deciding

# Comparison

Triggers:

- Decoupled from parent pipelines
- Pass variables downstream
- Child pipelines can use parallel

→ Scaling ✓
→ artifact handling?

Includes:

- Merging overwrites prior includes
- Setting variables possible but tedious
- No steps required for artifacts

→ Scaling ?
→ artifact handling ✓

# Agenda

1. Includes
2. Trigger
3. Difficulties
   - Scaling Includes
   - more than just CI
   - Artifacts
4. Deciding

# Specs to include pipeline multiple times

in `first`:

```yaml
spec:
  inputs:
    name:
      default: 'first'
---
stages:
  - first
  - second
  - third

include:
  - local: variables.yaml

doit$[[ inputs.name ]]:
  stage: $[[ inputs.name ]]
  script:
    - 'echo "${CI_PROJECT_NAME}, stage $[[
        inputs.name ]]: ${information}" >>
        data.html'
  artifacts:
    paths:
      - data.html
```

in `glue1-include`:

```yaml
---
include:
  - project: gent2024/first
    file: .gitlab-ci.yml
    ref: specs
    inputs:
      name: first
  - project: gent2024/first
    file: .gitlab-ci.yml
    ref: specs
    inputs:
      name: second
  - project: gent2024/first
    file: .gitlab-ci.yml
    ref: specs
    inputs:
      name: third
  - local: variables.yaml

variables:
  information:
    value: glue1 (.gitlab-ci.yml)
```

https://docs.gitlab.com/ee/ci/yaml/#spec

# Agenda

1. Includes
2. Trigger
3. Difficulties
   - Scaling Includes
   - more than just CI
   - Artifacts
4. Deciding

# Second brick

```yaml
spec:
  inputs:
    job-stage:
      default: 'first'
---
stages:
  - first
  - second
  - third

include:
  - local: variables.yaml

doit$[[ inputs.job-stage ]]:
  stage: $[[ inputs.job-stage ]]
  script:
    - ./doit.sh
  artifacts:
    paths:
      - data.html
```

- Logic externally (with respect to `.gitlab-ci.yml`)
  - Ansible roles
  - Terraform modules
  - Shell scripts
- Including: ✗
  → Only CI can be included
- Trigger: ✗
  → pass and retrieve artifact?

# Trigger artifact

.gitlab-ci.yml

```yaml
initialize:
  stage: initialize
  script:
    - git https://${CI_SERVER_HOST}/gent2024/
        second.git
  artifacts:
    paths:
      - second

doit:
  stage: doit
  variables:
    PARENT_PIPELINE_ID: ${CI_PIPELINE_ID}
  trigger:
    include:
      - local: child-pipeline.yaml
    strategy: depend
    forward:
      pipeline_variables: true
  parallel:
    matrix:
      - information:
          - A
          - B
```

child-pipeline.yaml

```yaml
include:
  - artifact: second/.gitlab-ci.yml
    pipeline: ${PARENT_PIPELINE_ID}
    job: initialize

doitfirst:
  needs:
    - pipeline: ${PARENT_PIPELINE_ID}
      job: initialize
  before_script:
    - cd second
  after_script:
    - cp second/data.html .
```

# Trigger artifact – why so complicated?

- child pipeline
  - artifact not yet there
- needs
  - requirement to obtain artifacts
- before and after script
  - directory shift
  - actions happen within `second`
  - artifacts are stored relative to `second`

# Agenda

1. Includes
2. Trigger
3. Difficulties
   - Scaling Includes
   - more than just CI
   - Artifacts
4. Deciding

# Upwards and downwards

- `needs` is your friend
- Passing artifacts “downwards” is easy
- Passing “upwards” from included pipelines is straigh forward, as well
- Open question: How to obtain artifacts from downstream pipelines (triggered pipelines)?

# Brickable: retrieve artifacts

```
retrieve_artifacts:
  stage: doit
  needs: ['doit']
  script:
    - >
        PIPELINE_IDS=$(curl -s --header "Authorization: Bearer ${PROJECT_TOKEN}"
        "${CI_SERVER_URL}/api/v4/projects/${CI_PROJECT_ID}/pipelines/${CI_PIPELINE_ID}/bridges"
        | jq -r ".[] | select(.name|test(\"doit\")) | .downstream_pipeline.id")
    - |
      OLDIFS=$IFS; IFS=$(echo -en "\n\b")
      for PREPARE_PIPELINE_ID in ${PIPELINE_IDS}
      do
        OUTPUT_JOB_ID=$(curl -s --header "Authorization: Bearer ${PROJECT_TOKEN}" \
        "${CI_SERVER_URL}/api/v4/projects/${CI_PROJECT_ID}/pipelines/${PREPARE_PIPELINE_ID}/jobs" \
        | jq ".[] | select(.name == \"doitfirst\") | .id")

        curl -s --header "Authorization: Bearer ${PROJECT_TOKEN}" --output artifacts.zip
        "${CI_SERVER_URL}/api/v4/projects/${CI_PROJECT_ID}/jobs/${OUTPUT_JOB_ID}/artifacts" \

        unzip artifacts.zip

        <DO SOMETHING TO ARTIFACTS>

        rm -rf artifacts.zip <ARTIFACTS>
      done
      IFS=$OLDIFS
  artifacts:
    <...>
```

# Agenda

1. Includes
2. Trigger
3. Difficulties
   - Scaling Includes
   - more than just CI
   - Artifacts
4. Deciding

# For all bricks

independent of method (includ, trigger, trigger artifact)

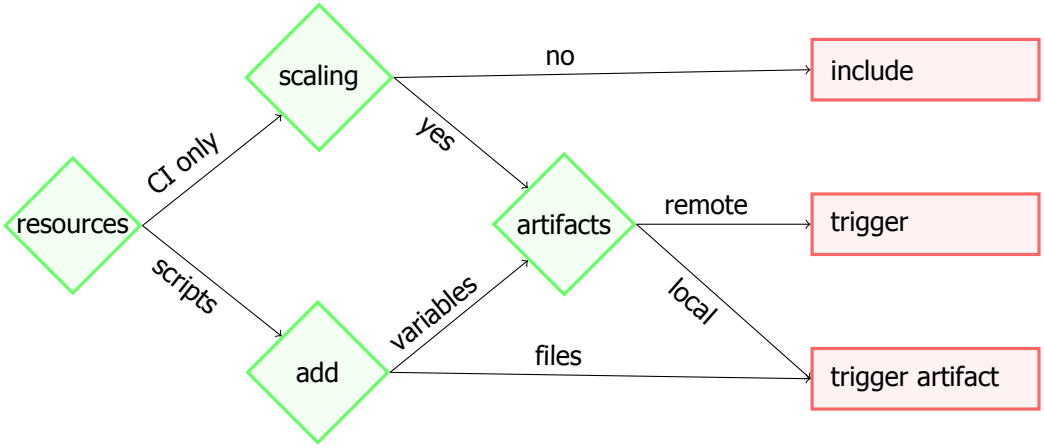
## Conventions

- no `before_script`
- no `after_script`
- well defined parametrization
  - `inventory.yaml`
  - `ssh_config`
- definition of child pipeline in `second`

# Matrix

| method | readability | easy to use | scales | artifacts | more than just CI |
|---|---|---|---|---|---|
| include | ✓ | ✓ | ✗✗ | local | ✗ |
| trigger | ✗✗ | ✓ | ✓ | remote | ✓ |
| trigger artifact | ✗ | ✗ | ✓ | local | ✓✓ |

# Summary


scaling
no
include
CI only
yes
resources
artifacts
remote
trigger
scripts
variables
local
add
files
trigger artifact