

gcil

Author: Adrian DC

Description: Launch .gitlab-ci.yml jobs locally

Date: 18/10/2026

Version: 13.2.0-1-ge2928df

► Usage

pypi

v13.2.0

python

3.10 | 3.11 | 3.12 | 3.13 | 3.14

downloads

1.7k/month

license

Apache License 2.0

pipeline

passed

bugs

0

code smells

0

coverage

100%

lines of code

5.6k

quality gate

passed

pre-commit

enabled

commitizen

friendly

gcil

enabled

mise

en place

radiandevcore

guidelines

pre-commit-crocodile

enabled

Launch .gitlab-ci.yml jobs locally, wrapped inside the specific images, with inplace project volume mounts and adaptive user selections

Documentation: <https://radiandevcore.gitlab.io/tools/gcil>

Package: <https://pypi.org/project/gitlabci-local/>

Table of contents

- [Purpose](#)
- [Preview](#)
- [Examples](#)
- [Usage](#)
- [Additional features in 'variables'](#)
- [Additional features in '.local'](#)
- [Job execution in native context](#)
- [Environment variables](#)
- [Supported container engines](#)
- [Supported systems](#)
- [Compatible projects](#)
- [Userspace available settings](#)
- [Environment available configurations](#)
- [Dependencies](#)
- [References](#)

Purpose

The main purpose of this project is to unify and enhance reliability of builds, tests or releases running on GitLab CI in a similar local context, by providing the simplicity of an interactive and automated terminal tool and avoiding code duplication (Makefile, Shell scripts, docker run, ...).

Rather than creating yet another standard, the `.gitlab-ci.yml` specification is the common and unique interface between GitLab CI and `gcil`.

Preview

A terminal window with a black background. The prompt 'adriandc@preview:~/gcil\$' is displayed in a green and red font. A white cursor is positioned at the end of the prompt.

```
adriandc@preview:~/gcil$
```

Examples

```
gcil                # Launch the jobs choices interactive menu
gcil -p             # Launch the jobs pipeline automatically
gcil -l             # Launch the job selection interactive menu
gcil -P 'Job 1'     # Launch the jobs choices menu with 'Job 1' preselected
gcil 'Dev'          # Launch jobs where the name contains a given string
gcil --debug 'Job 1' # Hold a finishing specific job for debugging
gcil --bash 'Job 1' # Prepare a bash environment for a specific job
gitlabci-local      # Shortcut alias to gcil
```

Usage

```
usage: gcil [-h] [--version] [--no-color] [--update-check] [--settings] [--set GROUP KEY VAL] [-p] [-q]
          [-c CONFIGURATION] [-B] [-A] [-C COMMANDS] [-n NETWORK] [-e ENV] [-E ENGINE] [-H] [--notify]
          [--privileged [BOOL]] [--random-paths] [-r] [-S] [--ssh [SSH_USER]] [-v VOLUME] [-w WORKDIR]
          [--bash | --debug] [--display] [--shell SHELL] [--all] [--configure] [--input KEY=VAL] [-f] [-i] [-m]
          [--no-console] [--no-git-safeties] [--no-script-fail] [-R] [--no-verbose] [--scripts] [-t TAGS]
          [-d | -s | -P | -l | --pull | --rmi] [--]
          [names ...]
```

gcil: Launch .gitlab-ci.yml jobs locally (aliases: gitlabci-local)

internal arguments:

```
-h, --help          # Show this help message
--version           # Show the current version
--no-color          # Disable colors outputs with 'NO_COLOR=1'
                    # (or default settings: [themes] > no_color)
--update-check      # Check for newer package updates
--settings          # Show the current settings path and contents
--set GROUP KEY VAL # Set settings specific 'VAL' value to [GROUP] > KEY
                    # or unset by using 'UNSET' as 'VAL'
```

pipeline arguments:

```
-p, --pipeline      # Automatically run pipeline stages rather than jobs
-q, --quiet         # Hide jobs execution context
-c CONFIGURATION    # Path to the .gitlab-ci.yml configuration file or folder
-B, --no-before     # Disable before_script executions
-A, --no-after      # Disable after_script executions
-C COMMANDS         # Run specific commands instead of "scripts" commands
-n NETWORK          # Configure the network mode used (or define CI_LOCAL_NETWORK)
                    # Choices: bridge, host, none. Default: bridge
-e ENV              # Define VARIABLE=value, pass VARIABLE or ENV file
-E ENGINE           # Force a specific engine (or define CI_LOCAL_ENGINE)
                    # Default list: auto,docker,podman
-H, --host          # Run jobs on the host rather than containers (or define CI_LOCAL_HOST)
--notify            # Enable host notifications of pipeline and jobs results
--privileged [BOOL] # Give extended privileges to the containers
                    # (Defaults to 'true', or define CI_LOCAL_PRIVILEGED)
--random-paths      # Mount random folder paths in the container
-r, --real-paths    # Mount real folder paths in the container (Linux / macOS only)
-S, --sockets       # Mount engine sockets for nested containers
                    # (Enabled by default with services: docker:*dind)
--ssh [SSH_USER]    # Bind SSH credentials to a container's user
-v VOLUME           # Mount VOLUME or HOST:TARGET in containers
-w WORKDIR          # Override the container's working path
```

debugging arguments:

```
--bash             # Prepare runners for manual bash purposes
--debug            # Keep runners active for debugging purposes
--display          # Enable host DISPLAY forwarding features
--shell SHELL      # Configure the default bash/debug shell entrypoint
```

jobs arguments:

```
--all              # Enable all jobs by default in selections
--configure        # Show interactive configurations for CI/CD inputs
--input KEY=VAL    # Define KEY=value for CI/CD inputs
-f, --force        # Force the action (use with --pull)
-i, --ignore-case  # Ignore case when searching for names
-m, --manual       # Allow manual jobs to be used
--no-console       # Disable console launch in bash/debug modes
                    # (or default settings: [runner] > no_console)
--no-git-safeties  # Disable automated Git safeties configuration
```

```
--no-script-fail    # (or default settings: [runner] > no_git_safeties)
                    # Fail on missing 'script' nodes of jobs
                    # (or default settings: [runner] > no_script_fail)
-R, --no-regex      # Disable regex search of names
--no-verbose        # Hide jobs verbose outputs
--scripts          # Dump parsed jobs entrypoint scripts
-t TAGS            # Handle listed tags as manual jobs
                    # Default list: deploy,local,publish

features arguments:
-d, --dump          # Dump parsed .gitlab-ci.yml configuration
-s, --select        # Force jobs selection from enumerated names
-P, --presets       # Select jobs with preset names
-l, --list          # Select one job to run (implies --manual)
--pull             # Pull container images from jobs
--rmi              # Delete container images from jobs

positional arguments:
--                 # Positional arguments separator (recommended)
names              # Names of specific jobs (or stages with --pipeline)
                    # Regex names is supported unless --no-regex is used
```

Additional features in 'variables'

`gcil` implements further support of most command-line parameters

using `variables:` values, either globally or specifically for each job:

```
variables:
  CI_LOCAL_AFTER: bool      # Enable or disable 'after_script' executions (see '-A')
  CI_LOCAL_BASH: bool       # Prepare runners for manual bash purposes (see '--bash')
  CI_LOCAL_BEFORE: bool     # Enable or disable 'before_script' executions (see '-B')
  CI_LOCAL_DEBUG: bool      # Keep runners active for debugging purposes (see '--debug')
  CI_LOCAL_DISPLAY: bool    # Enable host 'DISPLAY' forwarding features (see '--display')
  CI_LOCAL_ENGINE: str      # Force a specific engine (see '-E')
  CI_LOCAL_EXTENDS_INCOMPLETE: bool # Accept incomplete 'extends:' jobs for local use
  CI_LOCAL_HOST: bool       # Run jobs on the host rather than containers (see '--host')
  CI_LOCAL_MANUAL: bool     # Allow manual jobs to be used (see '--manual')
  CI_LOCAL_NETWORK: str     # Configure the network mode used (see '--network')
  CI_LOCAL_NO_CONSOLE: bool  # Disable console launch in bash/debug modes (see '--no-console')
  CI_LOCAL_NO_VERBOSE: bool  # Hide jobs verbose outputs (see '--no-verbose')
  CI_LOCAL_NOTIFY: bool     # Enable host notifications of pipeline and jobs results (see '--notify')
  CI_LOCAL_PRIVILEGED: bool  # Configure the privileged mode used (see '--privileged')
  CI_LOCAL_QUIET: bool      # Hide jobs execution context (see '-q')
  CI_LOCAL_RANDOM_PATHS: bool # Mount random folder paths in the container (see '--random-paths')
  CI_LOCAL_REAL_PATHS: bool  # Mount real folder paths in the container (see '-r')
  CI_LOCAL_SHELL: str       # Configure the default bash/debug shell entrypoint (see '--shell')
  CI_LOCAL_SOCKETS: bool    # Mount engine sockets for nested containers (see '-S')
  CI_LOCAL_SSH: bool|str    # Bind SSH credentials to a container's user (see '--ssh')
  CI_LOCAL_WORKDIR: str     # Override the container's working path (see '-w')
```

Additional features in '.local'

`gcil` implements further support of most command-line parameters

inside a `.local:` node to ease default parameters definitions.

Supported local values for a `.local` node are:

```
.local:
  all: bool      # Enable all jobs by default in selections (see '--all')
  env: list[str] # Define 'VARIABLE=value', pass 'VARIABLE' or 'ENV' file (see '-e')
  image: dict[str] # Override container image's 'name' and/or 'entrypoint'
  include: dict   # Map 'include: project:' names to local paths
  names: list[str] # Names of specific jobs (or stages with '--pipeline') (see 'names')
  no_regex: bool  # Disable regex search of names (see '--no-regex')
  pipeline: bool  # Automatically run pipeline stages rather than jobs (see '-p')
  tags: list[str] # Handle listed tags as manual jobs (see '--tags')
  variables: dict[str] # Define 'KEY: VALUE' variables for local jobs
  version: str    # Define a minimum version for 'gcil' recommended for this file
  volumes: dict[str] # Mount 'VOLUME' or 'HOST:TARGET' in containers (see '-v')
```

Examples for each of these can be found in the `local` unit tests: [tests/local](#) and [tests/includes](#)

Job execution in native context

`gcil` runs every jobs in the specified container image.

For specific local purposes where the native host context is wished, where the host tools, folders or credentials are required,

`image: local` can be used to run the scripts natively.

For specific purposes, the `image: local:quiet` variant can be used to enable the `quiet` option for specific jobs.

The `image: local:silent` variant extends the `quiet` option by also disabling the verbose script `set -x` line entry.

An example usage can be found in the local `Changelog` job: [.gitlab-ci.yml](#)

Environment variables

✎ Expand environment variables documentation

`gcil` uses the variables defined in `.gitlab-ci.yml` and parses the simple environment variables file named `.env`.

If specific environment variables are to be used in the job's container:

- `-e VARIABLE` : pass an environment variable
- `-e VARIABLE=value` : set a variable to a specific value
- `-e ENVIRONMENT_FILE` : parse a file as default variables

For example, `-e TERM=ansi` may enable colored terminal outputs.

The variable `CI_LOCAL` is automatically defined to `true` by `gcil` to allow specific conditions for local purposes in jobs' scripts.

The variable `CI_LOCAL_HOST` is automatically defined to `true` by `gcil` if running the job natively on the host (for example with `--host`)

The following variables are also defined by `gcil`:

- `CI_COMMIT_REF_NAME` : The branch or tag name for which project is built (GitLab CI)
- `CI_COMMIT_REF_SLUG` : `CI_COMMIT_REF_NAME` in lowercase, shortened to 63 bytes, and with everything except 0-9 and a-z replaced with -. No leading / trailing - (GitLab CI)
- `CI_COMMIT_SHA` : The commit revision for which project is built (GitLab CI)
- `CI_COMMIT_SHORT_SHA` : The first eight characters of `CI_COMMIT_SHA` (GitLab CI)
- `CI_COMMIT_TIMESTAMP` : The commit timestamp for which project is built (GitLab CI)
- `CI_PROJECT_NAME` : The name of the directory for the project (GitLab CI)
- `CI_PROJECT_NAMESPACE` : The project namespace (username or group name) of the job (GitLab CI)
- `CI_LOCAL_USER_HOST_GID` : The host user's group ID value
- `CI_LOCAL_USER_HOST_UID` : The host user's user ID value
- `CI_LOCAL_USER_HOST_USERNAME` : The host user's username value

Supported container engines

gcil currently supports these container engines:

- **Docker** : <https://docs.docker.com/get-docker/> (root daemon, as user or sudoer)
- **Podman** : <https://podman.io/getting-started/> (rootless or root CLI)

Supported systems

Supported Linux systems

Engines	Linux Mint, Ubuntu	CentOS	Others
Native (shell)	✓	✓	?
Docker (as user)	✓	✓	?
Docker (as root)	✓	✓	?
Podman (as user)	~	~	?
Podman (as root)	✓	✓	?

Supported macOS systems

Engines	macOS (10.14, 10.15, 11.0, ...)
Native (shell)	✓
Docker (as user)	?

 Supported Windows systems

▼

Engines	Windows 10 (1909, 2004, 20H2)	Others
Native (Command Prompt)	~	?
Native (Git Bash)	✓	?
Docker (Hyper-V)	✓	?
Docker (WSL 2)	✓	?

 Supported Android systems

▼

Engines	Android (7.0, 7.1, 8.0, 8.1, 9.0, 10, 11, ...)
Native (Termux)	✓

Compatible projects

Most GitLab CI projects should work with `gcil` without any local-specific changes.

However, if specific configurations like credentials, caches or user rights are needed, the `CI_LOCAL` variable can be used.

Projects compatible with `gcil` can use this badge to ease things for developers, both as an indicator and a documentation shortcut button :



Badge in Markdown

```
[![gcil](https://img.shields.io/badge/gcil-enabled-brightgreen?logo=gitlab)](https://radiandevcore.gitlab.io/tools/gcil)
```

Badge in HTML

```
<a href="https://radiandevcore.gitlab.io/tools/gcil"></a>
```

Userspace available settings

`gcil` creates a `settings.ini` configuration file in a userspace folder.

For example, it allows to change the default engines priority (`[engines]` > `engine`),
or to disable the automated updates daily check (`[updates]` > `enabled`)

The `settings.ini` file location and contents can be shown with the following command:

```
gcil --settings
```

Environment available configurations

`gcil` uses `colored` for colors outputs and `questionary` for interactive menus.

If colors of both outputs types do not match the terminal's theme,
an environment variable `NO_COLOR=1` can be defined to disable colors.

Dependencies

- [colored](#): Terminal colors and styles
 - [docker](#): Docker Engine API
 - [python-dotenv](#): Support for .env files parsing
 - [PyYAML](#): YAML parser and emitter for Python
 - [questionary](#): Interactive terminal user interfaces
 - [setuptools](#): Build and manage Python packages
 - [update-checker](#): Check for package updates on PyPI
-

References

- [.gitlab-ci.yml](#): GitLab CI/CD Pipeline Configuration Reference
- [commitizen](#): Simple commit conventions for internet citizens
- [git-cliff](#): CHANGELOG generator
- [gitlab-release](#): Utility for publishing on GitLab
- [OCI](#): Open Container Initiative
- [mkdocs](#): Project documentation with Markdown
- [mkdocs-coverage](#): Coverage plugin for mkdocs documentation
- [mkdocs-exporter](#): Exporter plugin for mkdocs documentation
- [mkdocs-material](#): Material theme for mkdocs documentation
- [mypy](#): Optional static typing for Python
- [pexpect-executor](#): Automate interactive CLI tools actions
- [pre-commit](#): A framework for managing and maintaining pre-commit hooks
- [pre-commit-crocodile](#): Git hooks intended for developers using pre-commit
- [PyPI](#): The Python Package Index
- [termtosvg](#): Record terminal sessions as SVG animations
- [Termux](#): Linux terminal emulator on Android
- [twine](#): Utility for publishing on PyPI
- [winpty](#): Windows PTY interface wrapper

gcil

Author: Adrian DC

Description: Launch .gitlab-ci.yml jobs locally

Date: 18/10/2026

Version: 13.2.0-1-ge2928df

► **GitLab CI**

pypi

v13.2.0

python

3.10 | 3.11 | 3.12 | 3.13 | 3.14

downloads

1.7k/month

license

Apache License 2.0

pipeline

passed

bugs

0

code smells

0

coverage

100%

lines of code

5.6k

quality gate

passed

pre-commit

enabled

commitizen

friendly

gcil

enabled

pre-commit-crocodile

enabled

Launch .gitlab-ci.yml jobs locally, wrapped inside the specific images,
with inplace project volume mounts and adaptive user selections.

Supported .gitlab-ci.yml features

Syntax references / .gitlab-ci.yml

```
# Global configurations

include: FILE_PATH
include:
  local: FILE_PATH
include:
  - local: FILE_PATH
include:
  - project: PROJECT_URL
    file: FILE_PATH
include:
  - project: PROJECT_URL
    file:
      - FILE_PATH_1
      - FILE_PATH_2

image: IMAGE_NAME
image:
  name: IMAGE_NAME
  entrypoint: ['COMMANDS']

services:
  - ...docker:dind
  - SERVICE_NAME
  - name: SERVICE_NAME
    alias: SERVICE_ALIAS

stages:
  - STAGE_NAMES

variables:
  - VARIABLES: VALUES
  - !reference [.path, to, YAML, node, variables]

# Global scripts

before_script:
  - COMMANDS
  - !reference [.path, to, YAML, node, before_script]

after_script:
  - COMMANDS
  - !reference [.path, to, YAML, node, after_script]

# Templates nodes

.TEMPLATES: &TEMPLATES
  KEYS: VALUES

# Job nodes

JOB_NAME:

# Job configurations

stage: STAGE_NAME

image: IMAGE_NAME
image:
  name: IMAGE_NAME
```

```
  entrypoint: ['COMMANDS']

services:
  - ...docker:dind
  - SERVICE_NAME
  - name: SERVICE_NAME
    alias: SERVICE_ALIAS

variables:
  VARIABLES: VALUES
  VARIABLE_PREFILLED:
    value: VALUE
    description: DESCRIPTION
  VARIABLE_REFERENCED: !reference [.path, to, YAML, node, variables, VARIABLE]
variables: !reference [.path, to, YAML, node, variables]

# Job templates

<<: *TEMPLATES
extends: TEMPLATE
extends:
  - TEMPLATES

# Job scripts

before_script:
  - COMMANDS
  - !reference [.path, to, YAML, node, after_script]

script:
  - COMMANDS
  - !reference [.path, to, YAML, node, script]

after_script:
  - COMMANDS
  - !reference [.path, to, YAML, node, after_script]

# Job executions

parallel:
  matrix:
    - VARIABLE1: VALUES
      VARIABLE2: VALUES
    - VARIABLE1: VALUES
      VARIABLE3: VALUES

retry: RETRY_COUNT
retry:
  max: RETRY_COUNT

tags:
  - MANUAL_TAGS

trigger: SIMPLE_TRIGGER (ignored)
trigger:
  COMPLEX_TRIGGER (ignored)

when: on_success / manual / on_failure / always

allow_failure: true / false
```

gcil

Author: Adrian DC

Description: Launch .gitlab-ci.yml jobs locally

Date: 18/10/2026

Version: 13.2.0-1-ge2928df

► Hooks

pypi

v13.2.0

python

3.10 | 3.11 | 3.12 | 3.13 | 3.14

downloads

1.7k/month

license

Apache License 2.0

pipeline

passed

🐛 bugs

0

🐼 code smells

0

🦋 coverage

100%

🦋 lines of code

5.6k

🦋 quality gate

passed

🔒 pre-commit

enabled

commitizen

friendly

🦊 gcil

enabled

🦊 pre-commit-crocodile

enabled

Launch `.gitlab-ci.yml` jobs locally, wrapped inside the specific images,
with inplace project volume mounts and adaptive user selections.

Table of contents

- [Using gcil hooks](#)
- [run-gcil-commit](#)
- [run-gcil-push](#)

Using gcil hooks

Sources / .pre-commit-config.yaml

```
# pre-commit configurations
default_install_hook_types:
  - pre-commit
  - pre-push
default_stages:
  - pre-commit
  - pre-push
minimum_pre_commit_version: 3.8.0

# gcil repositories
repos:

  # Repository: gcil
  - repo: https://gitlab.com/RadianDevCore/tools/gcil
    rev: X.Y.Z # Adapt to latest release tag
    hooks:
      - id: ...
```

run-gcil-commit

Run GitLab CI job with gcil upon commit and push :

• Sources / .pre-commit-config.yaml

```
# Repository: gcil
- repo: ...
  ...
  hooks:
    - id: run-gcil-commit
      name: Run GitLab CI job with gcil (adapt:job:name:1)
      description: Automatically run GitLab CI job with gcil (adapt:job:name:1)
      args:
        - 'adapt:job:name:1'
    - id: run-gcil-commit
      name: Run GitLab CI job with gcil (adapt:job:name:2)
      description: Automatically run GitLab CI job with gcil (adapt:job:name:2)
      args:
        - 'adapt:job:name:2'
```

run-gcil-push

Run GitLab CI job with gcil upon push only :

- Sources / .pre-commit-config.yaml

```
# Repository: gcil
- repo: ...
  ...
  hooks:
    - id: run-gcil-push
      name: Run GitLab CI job with gcil (adapt:job:name:1)
      description: Automatically run GitLab CI job with gcil (adapt:job:name:1)
      args:
        - 'adapt:job:name:1'
    - id: run-gcil-push
      name: Run GitLab CI job with gcil (adapt:job:name:2)
      description: Automatically run GitLab CI job with gcil (adapt:job:name:2)
      args:
        - 'adapt:job:name:2'
```

gcil

Author: Adrian DC

Description: Launch .gitlab-ci.yml jobs locally

Date: 18/10/2026

Version: 13.2.0-1-ge2928df

► Deprecations

pypi

v13.2.0

python

3.10 | 3.11 | 3.12 | 3.13 | 3.14

downloads

1.7k/month

license

Apache License 2.0

pipeline

passed

bugs

0

code smells

0

coverage

100%

lines of code

5.6k

quality gate

passed

pre-commit

enabled

commitizen

friendly

gcil

enabled

pre-commit-crocodile

enabled

Launch `.gitlab-ci.yml` jobs locally, wrapped inside the specific images,
with inplace project volume mounts and adaptive user selections.

Deprecated features

- **Since version 13.0.0 , the following `.local:` configurations are removed:**

- `.local: after` : Deprecated and removed, [documented in issue #292](#)
- `.local: bash` : Deprecated and removed, [documented in issue #292](#)
- `.local: before` : Deprecated and removed, [documented in issue #292](#)
- `.local: configurations` : Deprecated and removed, [documented in issue #292](#)
- `.local: debug` : Deprecated and removed, [documented in issue #292](#)
- `.local: default` : Deprecated and removed, [documented in issue #292](#)
- `.local: display` : Deprecated and removed, [documented in issue #292](#)
- `.local: engine` : Deprecated and removed, [documented in issue #292](#)
- `.local: manual` : Deprecated and removed, [documented in issue #292](#)
- `.local: network` : Deprecated and removed, [documented in issue #292](#)
- `.local: no_console` : Deprecated and removed, [documented in issue #292](#)
- `.local: no_verbose` : Deprecated and removed, [documented in issue #292](#)
- `.local: notify` : Deprecated and removed, [documented in issue #292](#)
- `.local: quiet` : Deprecated and removed, [documented in issue #292](#)
- `.local: random_paths` : Deprecated and removed, [documented in issue #292](#)
- `.local: real_paths` : Deprecated and removed, [documented in issue #292](#)
- `.local: shell` : Deprecated and removed, [documented in issue #292](#)
- `.local: sockets` : Deprecated and removed, [documented in issue #292](#)
- `.local: ssh` : Deprecated and removed, [documented in issue #292](#)
- `.local: workdir` : Deprecated and removed, [documented in issue #292](#)

- **Since version 12.0.0 , the following `.local:` configurations are deprecated:**

- `.local: after` : Use `CI_LOCAL_AFTER` variable globally or per job, [documented in issue #295](#)
- `.local: bash` : Use `CI_LOCAL_BASH` variable globally or per job, [documented in issue #295](#)

- `.local: before` : Use `CI_LOCAL_BEFORE` variable globally or per job, [documented in issue #295](#)
- `.local: configurations` : Deprecated and pending removal, [documented in issue #292](#)
- `.local: debug` : Use `CI_LOCAL_DEBUG` variable globally or per job, [documented in issue #295](#)
- `.local: default` : Deprecated and pending removal, [documented in issue #292](#)
- `.local: display` : Use `CI_LOCAL_DISPLAY` variable globally or per job, [documented in issue #295](#)
- `.local: engine` : Use `CI_LOCAL_ENGINE` variable globally or per job, [documented in issue #295](#)
- `.local: manual` : Use `CI_LOCAL_MANUAL` variable globally or per job, [documented in issue #295](#)
- `.local: no_console` : Use `CI_LOCAL_NO_CONSOLE` variable globally or per job, [documented in issue #295](#)
- `.local: no_verbose` : Use `CI_LOCAL_NO_VERBOSE` variable globally or per job, [documented in issue #295](#)
- `.local: notify` : Use `CI_LOCAL_NOTIFY` variable globally or per job, [documented in issue #295](#)
- `.local: quiet` : Use `CI_LOCAL_QUIET` variable globally or per job, [documented in issue #295](#)
- `.local: random_paths` : Use `CI_LOCAL_RANDOM_PATHS` variable globally or per job, [documented in issue #295](#)
- `.local: real_paths` : Use `CI_LOCAL_REAL_PATHS` variable globally or per job, [documented in issue #295](#)
- `.local: shell` : Use `CI_LOCAL_SHELL` variable globally or per job, [documented in issue #295](#)
- `.local: sockets` : Use `CI_LOCAL_SOCKETS` variable globally or per job, [documented in issue #295](#)
- `.local: ssh` : Use `CI_LOCAL_SSH` variable globally or per job, [documented in issue #295](#)
- `.local: workdir` : Use `CI_LOCAL_WORKDIR` variable globally or per job, [documented in issue #295](#)
- **Since version 11.0.0, the following `.local:` configurations are deprecated:**
 - `.local: network` : Use `CI_LOCAL_NETWORK` variable globally or per job, [documented in issue #286](#)
- **Versions before 10.0.0 were named `gitlabci-local` and not `gcil` yet,**
the tool's backwards compatibility remains, however settings have changed too,
use the `--settings` parameter to see the file and redo your changes
- **Versions before 4.6.0 used the now deprecated `PyInquirer` dependency,**
and due to its own old dependency to the `prompt-toolkit` package,
you should uninstall both packages first before updating

gcil