# gcil

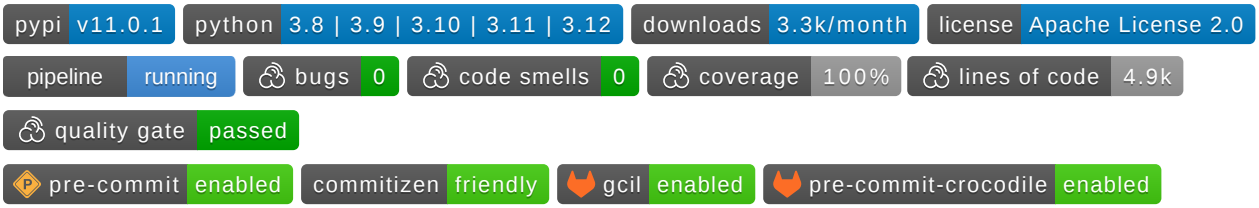
**Author:** Adrian DC

**Description:** Launch .gitlab-ci.yml jobs locally

**Date:** 15/03/2025

**Version:** 11.1.0

## ▸ Usage




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

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


---

## Examples

```
gcil                  # Launch the jobs choices interactive menu
gcil -p               # Launch the jobs pipeline automatically
gcil -l               # Launch the job selection interactive menu
gcil 'Dev'            # Launch jobs where the name contains a given string
gcil 'Job 1' --debug  # Hold a finishing specific job for debugging
gcil 'Job 1' --bash   # 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]
            [--random-paths] [-r] [-S] [--ssh [SSH_USER]] [-v VOLUME] [-w WORKDIR] [--bash | --debug] [--display]
            [--shell SHELL] [--all] [--defaults] [-f] [-i] [-m] [--no-console] [--no-git-safeties] [--no-script-fail]
            [-R] [--no-verbose] [--scripts] [-t TAGS] [-d | -s | -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 all jobs on the host rather than containers
  --notify             # Enable host notifications of pipeline and jobs results
  --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
  --defaults           # Use default variables for .local:configurations
  -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
                       # (or default settings: [runner] > no_git_safeties)
  --no-script-fail     # 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
  -l, --list           # Select one job to run (implies --manual)
  --pull               # Pull container images from all jobs
  --rmi                # Delete container images from all 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
```

# User configurations with '.local:configurations'

`gcil` implements support for specific user configurations
allowing simple and interactive local pipeline configurations.

Supported user configurations include `boolean`, `choice`, `input`, `yaml` and `json`.

Examples for each of these can be found in the `configurations` unit tests: tests/configurations

## 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_NETWORK: 'str'   # Configure the network mode used (see `--network`)
```

## 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:
  after: bool            # Enable or disable `after_script` executions (see `-A`)
  all: bool              # Enable all jobs by default in selections (see `--all`)
  bash: bool             # Prepare runners for manual bash purposes (see `--bash`)
  before: bool           # Enable or disable `before_script` executions (see `-B`)
  configurations: dict   # See [User configurations with '.local:configurations']](#user-configurations-with-
localconfigurations)
  debug: bool            # Keep runners active for debugging purposes (see `--debug`)
  defaults: bool         # Use default variables for `.local:configurations` (see `--defaults`)
  display: bool          # Enable host `DISPLAY` forwarding features (see `--display`)
  engine: str            # Force a specific engine (see `-E`)
  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
  manual: bool           # Allow manual jobs to be used (see `--manual`)
  names: list[str]       # Names of specific jobs (or stages with `--pipeline`) (see `names`)
  no_console: bool       # Disable console launch in bash/debug modes (see `--no-console`)
  no_regex: bool         # Disable regex search of names (see `--no-regex`)
  no_verbose: bool       # Hide jobs verbose outputs (see `--no-verbose`)
  notify: bool           # Enable host notifications of pipeline and jobs results (see `--notify`)
  pipeline: bool         # Automatically run pipeline stages rather than jobs (see `-p`)
  quiet: bool            # Hide jobs execution context (see `-q`)
  random_paths: bool     # Mount random folder paths in the container (see `--random-paths`)
  real_paths: bool       # Mount real folder paths in the container (see `-r`)
  shell: str             # Configure the default bash/debug shell entrypoint (see `--shell`)
  sockets: bool          # Mount engine sockets for nested containers (see `-S`)
  ssh: bool|str          # Bind SSH credentials to a container's user (see `--ssh`)
  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: bool          # Mount `VOLUME` or `HOST:TARGET` in containers (see `-v`)
  workdir: bool          # Override the container's working path (see `-w`)
```

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,
parses the simple environment variables file named `.env`
and the configurations selected through `.local:configurations`.

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 wiht `--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_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 and the `.local:` configurations 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"><img src="https://img.shields.io/badge/gcil-enabled-brightgreen?
logo=gitlab" alt="gcil" style="max-width:100%;"></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-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