pre-commit-crocodile

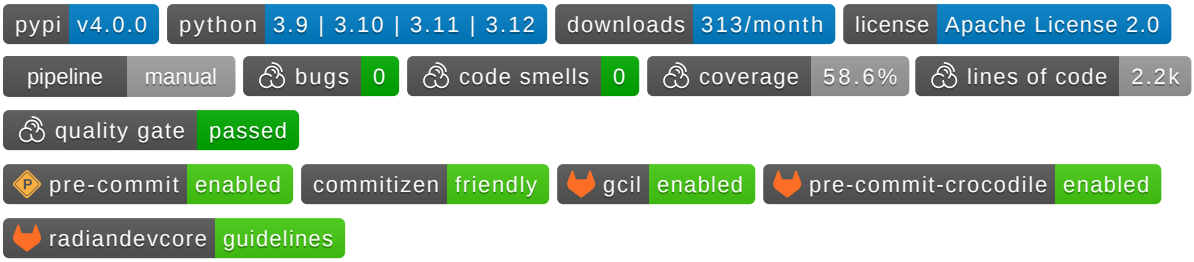
Author: Adrian DC

Description: Git hooks intended for developers using pre-commit

Date: 02/02/2025

Version: `4.0.0-7-g582f84f`

▸ Usage



Git hooks intended for developers using pre-commit and commitizen

Documentation: https://radiandevcore.gitlab.io/tools/pre-commit-crocodile
Package: https://pypi.org/project/pre-commit-crocodile/

Table of contents

Features

`pre-commit-crocodile` **uses the following features:**

- **CLI - pre-commit:** Automated Git hooks before commits and upon pushes
- **CLI - commitizen:** Commits tools and validation based upon conventional commits
- **Hooks - pre-commit-hooks:** Common `pre-commit` hooks useful for developers
- **Hooks -** `prepare-commit-msg` **:** Prepare commit message automatically based on changes

---

Preview

```
  --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'

modes arguments:
  -l, --list           List Git hooks installed in sources
  -i, --install        Install dependencies for pre-commit hooks
  -c, --configure      Update sources with hooks configurations
  -e, --enable         Enable pre-commit hooks
  -d, --disable        Disable pre-commit hooks
  -a, --autoupdate     Autoupdate pre-commit hooks
  -r, --run            Run pre-commit hooks

positional arguments:
  --                   Positional arguments separator (recommended)

adriandc@preview:~/pre-commit-crocodile$ pre-commit-crocodile --install

 pre-commit-crocodile (2.0.1)

 - Install dependencies: pre-commit

⣿ upgrading shared libraries
```

Usage

```
usage: pre-commit-crocodile [-h] [--version] [--no-color] [--update-check] [--settings] [--set GROUP KEY VAL]
                            [-l | -i | -c | -b | -e | -d | -a | -C | -r] [--config FOLDER | -D] [--stage STAGE] [--]

pre-commit-crocodile: Git hooks intended for developers using pre-commit

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'

modes arguments:
  -l, --list           # List Git hooks installed in sources
  -i, --install        # Install dependencies for pre-commit hooks
  -c, --configure      # Update sources with hooks configurations
  -b, --badges         # Update documentation with badges configurations
  -e, --enable         # Enable pre-commit hooks
  -d, --disable        # Disable pre-commit hooks
  -a, --autoupdate     # Autoupdate pre-commit hooks
  -C, --clean          # Clean pre-commit cached hooks
  -r, --run            # Run pre-commit hooks

configurations arguments:
  --config FOLDER      # Use configurations from a specific folder
  -D, --default        # Use global default configurations instead of sources
  --stage STAGE        # Run a specific pre-commit stage with --run
                       # (use 'list' to list supported stages)

positional arguments:
  --                   # Positional arguments separator (recommended)
```

Installation

```
{
  # Option 1: If using pipx
  if type pipx >/dev/null 2>&1; then
    pipx ensurepath
    pipx install pre-commit-crocodile
    pipx upgrade pre-commit-crocodile

  # Option 2: If using pip
  else
    sudo pip3 install pre-commit-crocodile
  fi
}
```

Compatibility

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

> 
>

Badge in Markdown

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

Badge in HTML

```
<a href="https://radiandevcore.gitlab.io/tools/pre-commit-crocodile"><img src="https://img.shields.io/badge/pre--
commit--crocodile-enabled-brightgreen?logo=gitlab" alt="pre-commit-crocodile" style="max-width:100%;"></a>
```

Projects with configurations | 

Install dependencies (once per user)

```
pre-commit-crocodile --install
```

Enable hooks for a project

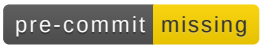
```
pre-commit-crocodile --enable
```

Manually run hooks of a project

```
pre-commit-crocodile --run
```

Disable hooks for a project

```
pre-commit-crocodile --disable
```

Projects without configurations | 

Import or refresh configurations

```
pre-commit-crocodile --configure
```

Projects maintenance | 

Update hooks automatically

```
pre-commit-crocodile --autoupdate
```

Cleanup hooks cache

```
pre-commit-crocodile --clean
```

Dependencies

- colored: Terminal colors and styles
- commitizen: Simple commit conventions for internet citizens
- pre-commit: A framework for managing and maintaining pre-commit hooks
- pre-commit-crocodile: Git hooks intended for developers using pre-commit
- setuptools: Build and manage Python packages
- update-checker: Check for package updates on PyPI

References

- .gitlab-ci.yml: GitLab CI/CD Pipeline Configuration Reference
- conventionalcommits: Conventional Commits specification for commit messages
- gcil: Launch .gitlab-ci.yml jobs locally
- git-cliff: CHANGELOG generator
- gitlab-release: Utility for publishing on GitLab
- 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
- PyPI: The Python Package Index
- termtosvg: Record terminal sessions as SVG animations
- twine: Utility for publishing on PyPI

pre-commit-crocodile

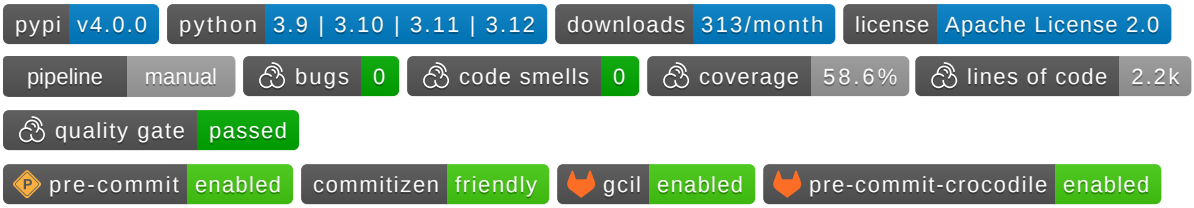
Author: Adrian DC

Description: Git hooks intended for developers using pre-commit

Date: 02/02/2025

Version: `4.0.0-7-g582f84f`

▸ Commits



Git hooks intended for developers using pre-commit and commitizen.

Table of contents

Commits commands

- `git commit` **:** Create a commit with the title prepared by `prepare-commit-message`
- `git commit -s` **:** Create a signed commit with the title prepared by `prepare-commit-message`
- `cz commit` **:** Create a commit interactively using `commitizen` by answering specific questions

---

Conventional Commits - Customized specification

The commit contains the following structural elements:

Commit type

- `fix` **: Bug** or **issues fixes**
- `feat` **:** New or extended **feature**
- `chore` **:** Changes to **non-production** code
- `docs` **: Documentation** only changes
- `style` **: Cosmetic** changes only (white-space, formatting, etc)
- `refactor` **:** Code changes that **neither** fix a **bug** nor add a **feature**
- `perf` **:** Code change that **improve performance**
- `security` **:** Changes resolve **security** related **issues**
- `test` **:** Adding missing or correcting existing **tests**
- `build` **:** Changes to the **build** system or dependencies (**example:** `makefile, pip, docker`)
- `ci` **:** Changes to **CI** configuration files and scripts (**example:** `gitlab-ci`)
- `improvement` **:** Changes that **improve** sources without any impact
- `wip` **: Temporary** commits meant for **local** developments and rebases

Commit scope

A scope has to be provided to a commit's type, to provide additional contextual information, such as the **changed file**, the containing folder or a globally **modified feature**, and is contained **within parenthesis**, e.g. `feat(parser): add ability to parse arrays` .

Commit breaking changes

`BREAKING CHANGE:` A commit with `!` before the `:` or that has the text `BREAKING CHANGE:` at the beginning of its optional body or footer section introduces a breaking API change (correlating with MAJOR in **semantic versioning**).

A `BREAKING CHANGE` can be part of commits of any type.

Commit syntax

Notice these types are not mandated by the **conventional commits** specification, and have no implicit effect in **semantic versioning** (unless they include a `BREAKING CHANGE`).

```
<type>(mandatory scope)[optional !]: <description>

[optional body]

[optional footer]
```

Commit example

```
feat(scope): implement feature ABC in sources

Issue: #123
Details: Details about the changes
---

Signed-off-by: Firstname LASTNAME <firstname.lastname@example.com>
```

pre-commit-crocodile

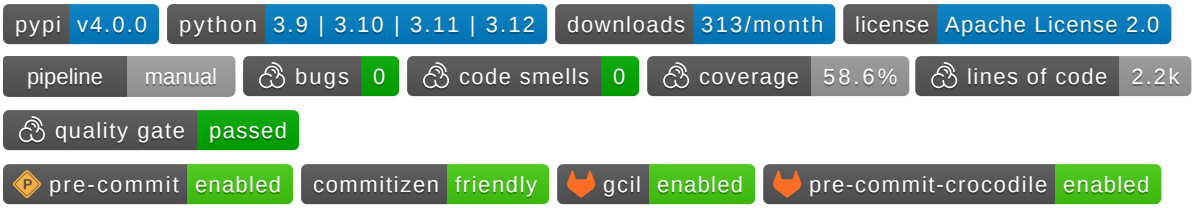
Author: Adrian DC

Description: Git hooks intended for developers using pre-commit

Date: 02/02/2025

Version: `4.0.0-7-g582f84f`

▸ Hooks



Git hooks intended for developers using pre-commit and commitizen.

Table of contents

Using pre-commit-crocodile hooks

Sources / .pre-commit-config.yaml

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

# pre-commit repositories
repos:

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

`prepare-commit-message`

Automatically prepare the commit message based on changed sources before manual edition :

- **Sources / .pre-commit-config.yaml**

```yaml
# Repository: pre-commit-crocodile
- repo: ...
  ...
  hooks:
    - id: prepare-commit-message
```

- **Launched automatically by Git** upon `prepare-commit-msg` stage using `pre-commit`
- Parse `Changes to be committed:` **to automatically prepare a** `type(scope): ...` **title**
- Commit messages implementation derived from **Conventional Commits specifications**
- **Automatically detect and prepare templates like** `ci(gitlab-ci):` **,** `docs(readme):` **,** `build(makefile):` **or** `docs(changelog):`
- **Automatically evaluate sources specific scopes, for example files or folders under** `src/` **, or recipe folders in Yocto** `recipes-.../` **sources**
- Message body is automatically inserted to help developers document their commits or link to a related issue
- Commits with sign-off messages automatically receive a `---` separator for readability on GitLab

---

`check-yaml-ruamel-pure`

Check YAML files using ruamel.yaml pure Python implementation :

- **Sources / .pre-commit-config.yaml**

```yaml
# Repository: pre-commit-crocodile
- repo: ...
  ...
  hooks:
    - id: check-yaml-ruamel-pure
```

- **Specifically created to use the pure Python implementation of** `ruamel.yaml`
- Recommended here by the maintainer (Anthon van der Neut)
- **Use this hook if** `check-yaml` **raises these parsing failures :**

```
ruamel.yaml.scanner.ScannerError: while scanning a plain scalar, found unexpected ':'
```

- **Drop-in alternative to the** `check-yaml` original hook's `--unsafe` mode

pre-commit-crocodile

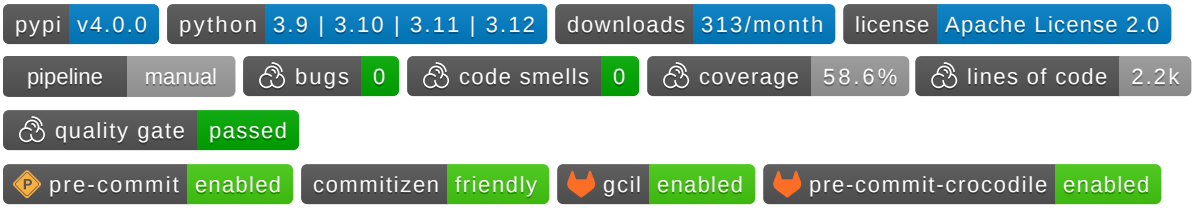
Author: Adrian DC

Description: Git hooks intended for developers using pre-commit

Date: 02/02/2025

Version: `4.0.0-7-g582f84f`

▸ commitizen



Git hooks intended for developers using pre-commit and commitizen.

Configurations for commitizen

Sources / .cz.yaml

```
---
commitizen:

  # commitizen configurations
  always_signoff: true
  bump_message: 'docs(changelog): regenerate release tag changes history'
  changelog_incremental: false
  major_version_zero: true
  retry_after_failure: false
  tag_format: $version
  update_changelog_on_bump: false
  use_shortcuts: true
  version_provider: scm
  version_scheme: pep440

  # commitizen rules
  #  - Based upon 'cz_conventional_commits' original rules for Conventional Commits
  #  - Improved with 'Issue:' detailed messages examples and 'Signed-off-by' footer
  #  - Commit scope turned mandatory by the rules
  #  - Commit scope refactored with support for spaces and comma
  #  - Breaking change reimplemented with the standard '!' scope suffix
  #  - Questions simplified for professional usage
  #  - Type 'security' implemented for security related commits
  #  - Accept temporary commits starting with 'wip' or 'WIP'
  name: cz_customize
  customize:
    example: |-
      feat(scope): implement feature ABC in sources

      Issue: #123
      Details: Details about the changes
      ---

      Signed-off-by: Firstname LASTNAME <firstname.lastname@example.com>
    info: |

      ## Conventional Commits - Customized specification

      The commit contains the following structural elements:

      ### Commit type

      - fix: Bug or issues fixes
      - feat: New or extended feature
      - chore: Changes to non-production code
      - docs: Documentation only changes
      - style: Cosmetic changes only (white-space, formatting, etc)
      - refactor: Code changes that neither fix a bug nor add a feature
      - perf: Code change that improve performance
      - security: Changes resolve security related issues
      - test: Adding missing or correcting existing tests
      - build: Changes to the build system or dependencies (example: makefile, pip, docker)
      - ci: Changes to CI configuration files and scripts (example: gitlab-ci)
      - improvement: Changes that improve sources without any impact
      - wip: Temporary commits meant for local developments and rebases

      ### Commit scope

      A scope has to be provided to a commit's type, to provide additional contextual information,
```

```
      such as the changed file, the containing folder or a globally modified feature,
      and is contained within parenthesis, e.g. feat(parser): add ability to parse arrays.

      ### Commit breaking changes

      BREAKING CHANGE: A commit with '!' before the ':' or that has the text BREAKING CHANGE:
      at the beginning of its optional body or footer section introduces a breaking API change
      (correlating with MAJOR in semantic versioning).

      A BREAKING CHANGE can be part of commits of any type.

      ### Commit syntax

      Notice these types are not mandated by the conventional commits specification,
      and have no implicit effect in semantic versioning (unless they include a BREAKING CHANGE).

      <type>(mandatory scope)[optional !]: <description>

      [optional body]

      [optional footer]
    message_template: "\
      {{prefix}}\
      ({{scope}})\
      {% if is_breaking_change %}!{% endif %}\
      : \
      {{subject}}\
      {% if body or footer %}\n{% endif %}\
      {% if body %}\n{{body}}{% endif %}\
      {% if footer %}\n{{footer}}{% endif %}\
      {% if body or footer %}\n---{% endif %}\
    "
    questions:
      - type: list
        name: prefix
        message: 'Type of the change :'
        choices:
          - value: fix
            name: 'fix: Bug or issues fixes'
            key: x
          - value: feat
            name: 'feat: New or extended feature'
            key: f
          - value: chore
            name: 'chore: Changes to non-production code'
            key: h
          - value: docs
            name: 'docs: Documentation only changes'
            key: d
          - value: style
            name: 'style: Cosmetic changes only (white-space, formatting, etc)'
            key: s
          - value: refactor
            name: 'refactor: Code changes that neither fix a bug nor add a feature'
            key: r
          - value: perf
            name: 'perf: Code change that improve performance'
            key: p
          - value: security
            name: 'security: Changes that improve sources without any impact'
```

```
            key: e
          - value: test
            name: 'test: Adding missing or correcting existing tests'
            key: t
          - value: build
            name: 'build: Changes to the build system or dependencies (example: makefile, pip, docker)'
            key: b
          - value: ci
            name: 'ci: Changes to CI configuration files and scripts (example: gitlab-ci)'
            key: c
          - value: improvement
            name: 'improvement: Changes that improve sources without any impact'
            key: i
      - type: input
        name: scope
        message: 'Scope of the change :'
        filter: 'required_validator_scope'
        default: ''
      - type: input
        name: subject
        message: 'Title of the commit (starting in lower case and without period) :'
        filter: 'required_validator_title_strip'
        default: ''
      - type: input
        name: body
        message: 'Additional contextual message (Empty to skip) :'
        default: 'Issue: #...'
        filter: 'multiple_line_breaker'
      - type: input
        name: footer
        message: 'Footer information and references (Empty to skip) :'
        default: ''
      - type: confirm
        message: 'BREAKING CHANGE to warn ?'
        name: is_breaking_change
        default: False
    schema: |-
      <type>(<scope>): <subject>
      <BLANK LINE>
      <body>
      <BLANK LINE>
      <footer>
    schema_pattern: "\
      (?s)\
      ^([Ww][Ii][Pp]).*$\
      |\
      (fix|feat|chore|docs|style|refactor|perf|test|build|ci|improvement|security|revert)\
      (\\([^\\n\\r]+\\))\
      !?\
      :\
       \
      ([^\\n\\r]+)\
      ((\\n\\n.*)|(\\s*))?\
      (\\n[^\\n\r]+-by:.*)*\
      $\
    "

  # commitizen styles
  style:
    - ["qmark", "fg:#FF9D00 bold"]
```

```
    - ["question", "bold"]
    - ["answer", "fg:#FF9D00 bold"]
    - ["pointer", "fg:#FF9D00 bold"]
    - ["highlighted", "fg:#FF9D00 bold"]
    - ["selected", "fg:#CC5454"]
    - ["separator", "fg:#CC5454"]
    - ["instruction", ""]
    - ["text", ""]
    - ["disabled", "fg:#858585 italic"]
```

pre-commit-crocodile

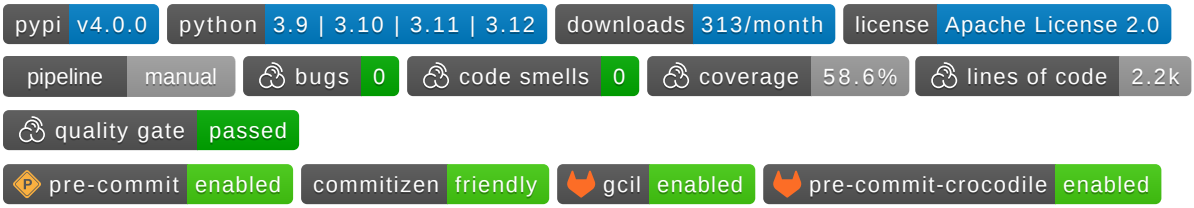
Author: Adrian DC

Description: Git hooks intended for developers using pre-commit

Date: 02/02/2025

Version: `4.0.0-7-g582f84f`

▸ pre-commit



Git hooks intended for developers using pre-commit and commitizen.

Configurations for pre-commit

Sources / .pre-commit-config.yaml

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

# pre-commit exclusions
exclude: >
  (?x)(
    ^archives/|
    \.drawio$|
    \.patch$|
    \.svg$|
    \.vhd$|
    \.xci$|
    \.xdc$|
    eicar\..*$|
    CHANGELOG.md$|
    COPYING.*$|
    LICENSE.*$|
    README\.md$
  )

# pre-commit repositories
repos:

  # Repository: pre-commit
  - repo: meta
    hooks:
      - id: check-hooks-apply
      # - id: check-useless-excludes

  # Repository: pre-commit-hooks
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.6.0
    hooks:
      - id: check-added-large-files
        args: ['--maxkb=500']
      - id: check-case-conflict
        exclude: >
          (?x)(
            \.nmconnection$
          )
      - id: check-executables-have-shebangs
      - id: check-json
      - id: check-merge-conflict
      # - id: check-toml
      # - id: check-xml
      - id: check-yaml
        args: ['--unsafe']
      - id: destroyed-symlinks
        stages: [commit, push, manual] # Issue: https://github.com/pre-commit/pre-commit-hooks/pull/1085
```

```
      # - id: detect-private-key
      - id: double-quote-string-fixer
      - id: end-of-file-fixer
      - id: mixed-line-ending
      - id: trailing-whitespace
        args: ['--markdown-linebreak-ext=md']

  # Repository: pre-commit-hooks
  - repo: https://github.com/pre-commit/pygrep-hooks
    rev: v1.10.0
    hooks:
      - id: python-check-blanket-type-ignore
      - id: python-no-eval

  # Repository: commitizen
  - repo: https://github.com/commitizen-tools/commitizen
    rev: v3.29.0
    hooks:
      - id: commitizen
      - id: commitizen-branch
        entry: cz
        args: [--no-raise, '3', check, --rev-range, origin/HEAD..HEAD]
        stages:
          - push

  # Repository: pre-commit-crocodile
  - repo: local
    hooks:
      - id: check-yaml-ruamel-pure
        name: Check YAML (ruamel.yaml pure Python)
        entry: python ./src/hooks/check_yaml_ruamel_pure.py
        types: [yaml]
        args: []
        language: python
        description: Check YAML (ruamel.yaml pure Python implementation)
        stages:
          - commit
          - push
          - manual
        additional_dependencies:
          - ruamel.yaml>=0.18.6
      - id: prepare-commit-message
        name: Prepare commit message automatically
        entry: python ./src/hooks/prepare_commit_message.py
        args: []
        language: python
        description: Automatically prepare the commit message for manual edition
        stages:
          - prepare-commit-msg
        additional_dependencies:
          - PyYAML>=6.0

  # Repository: gcil
  - repo: https://gitlab.com/RadianDevCore/tools/gcil
    rev: 9db7ca21
    hooks:
      - id: run-gcil-push
        name: Run GitLab CI job with gcil (codestyle)
        description: Automatically run GitLab CI job with gcil (codestyle)
        args:
```

```
        - 'codestyle'
    - id: run-gcil-push
      name: Run GitLab CI job with gcil (lint)
      description: Automatically run GitLab CI job with gcil (lint)
      args:
        - 'lint'
    - id: run-gcil-push
      name: Run GitLab CI job with gcil (typings)
      description: Automatically run GitLab CI job with gcil (typings)
      args:
        - 'typings'
```

pre-commit-crocodile

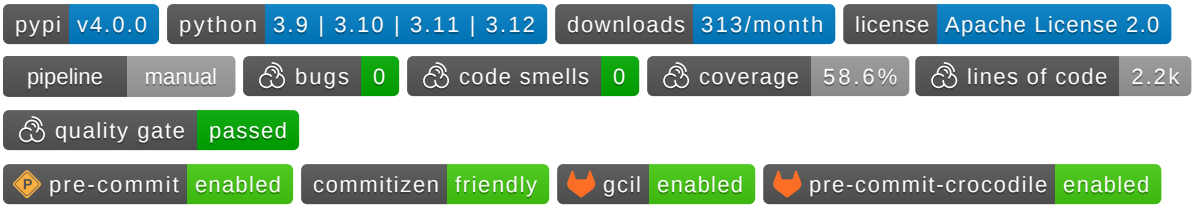
Author: Adrian DC

Description: Git hooks intended for developers using pre-commit

Date: 02/02/2025

Version: `4.0.0-7-g582f84f`

▸ Components



Git hooks intended for developers using pre-commit and commitizen.

CI/CD components

GitLab CI/CD Catalog: https://gitlab.com/explore/catalog/RadianDevCore/tools/pre-commit-crocodile

---

`commits` - GitLab CI job to validate newly pushed commits

Validate commits added on a branch or for a merge request :

- **Show commits specifications** and syntax examples
- **Check commits automatically** with `commitizen` configurations
- **Run pre-commit checks** on all files
- **Deny `WIP` commits** on GitLab branches

Sources / .gitlab-ci.yml

```yaml
include:
  - component: gitlab.com/RadianDevCore/tools/pre-commit-crocodile/commits@X.Y.Z # Adapt to latest release tag
    inputs:
      stage: prepare

stages:
  - prepare
  - ...
```

pre-commit-crocodile

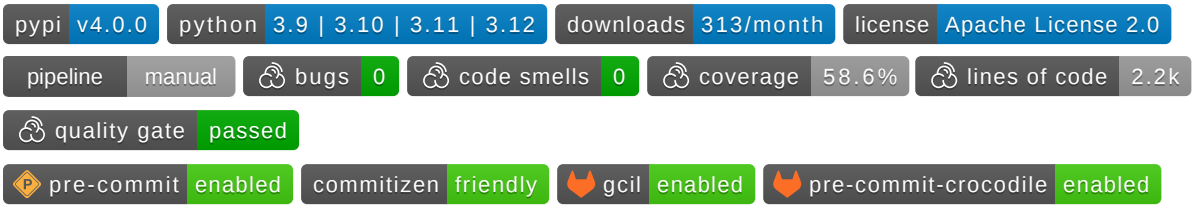
Author: Adrian DC

Description: Git hooks intended for developers using pre-commit

Date: 02/02/2025

Version: `4.0.0-7-g582f84f`

▸ Settings



Git hooks intended for developers using pre-commit and commitizen.

Userspace available settings

`pre-commit-crocodile` creates a `settings.ini` configuration file in a userspace folder.

For example, it allows to disable the automated updates daily check (`[updates] > enabled`)

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

```
pre-commit-crocodile --settings
```

Environment available configurations

`pre-commit-crocodile` 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.

pre-commit-crocodile