# dotenv-linter

*Release 0.5.0*

**wemake.services**

**Apr 29, 2024**

# CONTENTS

Simple linter for `.env` files.


While `.env` files are very simple it is required to keep them consistent. This tool offers a wide range of consistency rules and best practices.

And it integrates perfectly to any existing workflow.

Read the announcing post.

CHAPTER

# ONE

# INSTALLATION AND USAGE

```
pip install dotenv-linter
```

And then run it:

```
dotenv-linter .env .env.template
```

See Usage section for more information.

CHAPTER

TWO

# EXAMPLES

There are many things that can go wrong in your `.env` files:

```
# Next line has leading space which will be removed:
 SPACED=

# Equal signs should not be spaced:
KEY = VALUE

# Quotes won't be preserved after parsing, do not use them:
SECRET="my value"

# Beware of duplicate keys!
SECRET=Already defined ;(

# Respect the convention, use `UPPER_CASE`:
kebab-case-name=1
snake_case_name=2
```

And much more! You can find the full list of violations in our docs.

CHAPTER

# THREE

# PRE-COMMIT HOOKS

`dotenv-linter` can also be used as a pre-commit hook. To do so, add the following to the `.pre-commit-config.yaml` file at the root of your project:

```yaml
repos:
  - repo: https://github.com/wemake-services/dotenv-linter
    rev: 0.2.0  # Use the ref you want to point at
    hooks:
      - id: dotenv-linter
```

For the more detailed instructions on the pre-commit tool itself, please refer to its website.

CHAPTER

# FOUR

# GRATIS

Special thanks goes to Ignacio Toledo for creating an awesome logo for the project.

## 4.1 Usage

### 4.1.1 dotenv-linter

Main entrypoint to the app.

Runs `lint` command by default if nothing else is not specified. Runs `--version` subcommand if this option is provided.

```
dotenv-linter [OPTIONS] COMMAND [ARGS]...
```

#### Options

**--version**

#### lint

Runs linting process for the given files.

```
dotenv-linter lint [OPTIONS] FILES...
```

#### Arguments

**FILES**

Required argument(s)

## 4.2 Violations

### 4.2.1 Parsing

Different error that might happen during file parsing phase.

**final class ParsingViolation**(*node=None*, *text=None*)

Bases: BaseFileViolation

Indicates that given file can not be correctly parsed.

This may include: 1. Incorrect OS behavior 2. Incorrect syntax inside this file 3. Errors in our grammar definition 4. Our internal errors

Added in version 0.1.0.

### 4.2.2 Names

Rules that define how names should be defined.

**final class SpacedNameViolation**(*node*, *text*)

Bases: BaseFSTViolation

Restricts to use duplicate names variables.

**Reasoning:**
This spaces will be removed by the parsing mechanism, but they might cause some confusion to users.

**Solution:**
Remove leading spaces.

Example:

```
# Correct:
SOME_KEY=1

# Wrong:
    SOME_KEY=1
```

Added in version 0.1.0.

**final class IncorrectNameViolation**(*node*, *text*)

Bases: BaseFSTViolation

Restricts to use restricted symbols to define names.

**Reasoning:**
By convention we can only use letters, numbers, and underscores to define dotenv variables. Moreover, variables can not start with numbers.

**Solution:**
Refactor your file to contain only allowed characters.

Example:

```
# Correct:
SOME_KEY=1
```

(continues on next page)

(continued from previous page)

```
# Wrong:
SOME-KEY=1
```

Added in version 0.1.0.

**final class DuplicateNameViolation**(*node*, *text*)

Bases: BaseFSTViolation

Restricts to use duplicate names variables.

**Reasoning:**
There is no need to create duplicate variables inside your dotenv file. Since it will be implicitly overridden by the parsing mechanism.

**Solution:**
Remove one of the duplicate variables.

Example:

```
# Correct:
SOME_KEY=1
OTHER_KEY=2

# Wrong:
SOME_KEY=1
SOME_KEY=2
```

Added in version 0.1.0.

**final class RawNameViolation**(*node*, *text*)

Bases: BaseFSTViolation

Restricts to use raw names without equal sign or value.

**Reasoning:**
It does not make any sense to just state some names. It might also break some `.env` parsers.

**Solution:**
Append equal sign to it. So, this would became a declaration of an empty variable. You can also add a value if it makes sense.

Example:

```
# Correct:
KEY=1
OTHER=

# Wrong:
KEY
```

Added in version 0.1.0.

**final class ReservedNameViolation**(*node*, *text*)

Bases: BaseFSTViolation

Restricts to use of blacklisted names.

**Reasoning:**
It does not make any sense to use such names.

**Solution:**
Change such names. Or, Add _ at the end, to distinguish it from reserved ones.

Example:

```
# Wrong:
DJANGO_ENV=some_value
```

Added in version 0.2.0.

### 4.2.3 Assigns

Rules that define how assigns should be made.

`final class SpacedAssignViolation`(*node*, *text*)

Bases: `BaseFSTViolation`

Restricts to write = signs with extra spaces.

**Reasoning:**
Valid `shell` syntax requires to write assigns without any spaces.

**Solution:**
Remove any spaces between the = char.

Example:

```
# Correct:
KEY=1
OTHER=

# Wrong:
KEY = 1
OTHER =
```

Added in version 0.1.0.

### 4.2.4 Values

Rules about writing correct dotenv values.

By convention we do not print values to the output. Since they might contain private values.

`final class SpacedValueViolation`(*node*, *text*)

Bases: `BaseFSTViolation`

Restricts to write values with trailing spaces.

**Reasoning:**
These spaces are not guaranteed to be preserved. So, it is better not to rely on them.

**Solution:**
Remove trailing spaces from the value.

Added in version 0.1.0.

**final class QuotedValueViolation**(*node*, *text*)

Bases: BaseFSTViolation

Restricts to quoted values.

**Reasoning:**
Dotenv parser usually strips quotes away, so it is hard to say whether these quotes will stay on a final value, or not.

**Solution:**
Remove any quotes from the value.

Example:

```
# Correct:
KEY=1

# Wrong:
KEY="1"
```

Added in version 0.1.0.

### 4.2.5 Comments

Rules that define how comments should be written.

**final class SpacedCommentViolation**(*node*, *text*)

Bases: BaseFSTViolation

Restricts to write comment with leading or trailing spaces.

**Reasoning:**
These spaces are meaningless and will be removed. So, why would you want to have them?

**Solution:**
Remove leading or trailing spaces from the comment body.

Added in version 0.1.0.

## 4.3 Docker

We have an existing official image on DockerHub.

### 4.3.1 Usage

You can can use it like so:

```
docker pull wemakeservices/dotenv-linter
docker run --rm wemakeservices/dotenv-linter .env
```

Make sure to place proper config file and mount it with the source code like so:

```
docker run --rm wemakeservices/dotenv-linter -v `pwd`:/code /code
```

You can also use this image with Gitlab CI or any other container-based CIs.

### 4.3.2 Further reading

- Official 'docker run' docs
- Official GitlabCI docs

## 4.4 Github Actions

Good news: we ship pre-built Github Action with this project.

You can use it from the Github Marketplace:

```
- name: dotenv-linter
  uses: wemake-services/dotenv-linter
```

You can also specify any version starting from `0.1.5` instead of the default `latest` tag.

### 4.4.1 Inputs

#### reporter

We support three reporting options:

- `terminal` (default one) when we just dump the output into Action's logs. Is the easiest one to setup, that's why we use it by default
- `github-pr-review` (recommended) when we use inline comments inside code reviews
- `github-pr-check` when we use Github Checks for the output

Take a note that `github-pr-review` and `github-pr-check` requires `GITHUB_TOKEN` environment variable to be set.

For example, that's how `github-pr-reviews` can be set up:

```
- name: dotenv-linter
  uses: wemake-services/dotenv-linter
  with:
    reporter: 'github-pr-review'
  env:
    GITHUB_TOKEN: ${{ secrets.github_token }}
```

#### options

We also support custom CLI `options` to be specified, they are exactly match anything that can be provided to `dotenv-linter` itself:

```
- name: dotenv-linter
  uses: wemake-services/dotenv-linter
  with:
    options: './conf/.env ./conf/.env.docker'
```

### 4.4.2 Outputs

We also support `outputs` from the spec, so you can later pass the output of `dotenv-linter` to somewhere else.

```
- name: dotenv-linter
  uses: wemake-services/dotenv-linter
- name: Custom Action
  runs: echo "{{ steps.dotenv-linter.outputs.output }}"
```

CHAPTER

FIVE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## d

# INDEX

## Symbols



## D



## F



## I



## M



## P



## Q



## R



## S