compile-python-requirements-action

Release 0.1

Artur Barseghyan <artur.barseghyan@gmail.com>

Dec 30, 2023

CONTENTS

Compile Python requirement files for MacOs, Linux or Windows.

CHAPTER

ONE

FEATURES

- Compile Python requirement files (from requirements.in or pyproject.toml) for MacOs, Linux or Windows.
- Create PRs with the changes.
- Create Artifacts with compiled requirements.

CHAPTER

TWO

DISCLAIMER

This project will be irrelevant for you if:

- Your **entire team** uses Docker for development.
- Or you are the **sole developer** of a project.
- Or your entire team uses **the same** operating system (MacOs, Linux or Windows).

If none of the statements above qualifies, read further.

CHAPTER

THREE

WHY

When a single project is being worked on by multiple developers, it's useful to streamline the installation process and ensure everyone is using exactly the same package versions.

It's a good practice to compile your input requirements and build your production environment from compiled requirements to prevent unpleasant surprises.

In many cases, pip-tools is your best friend, however, what pip-tools doesn't do is compiling requirements for multiple platforms. It can only compile requirements for the platform it's being executed on.

Software engineering is multi-diverse. Some may find Docker an essential tool, some others won't even bother using it and would prefer to work in a virtual environment.

And then it's your job to make sure everyone is working efficiently and does not accidentally break the project. If a Linux user makes a change in input requirements, you want to make sure requirements are properly compiled for all designated systems so that when a MacOs user pulls the changes and runs the installation script, he immediately has the project up-and-running with properly synchronized package versions.

CHAPTER

FOUR

CONFIGURATION

4.1 inputs

4.1.1 *input-file*

Required The path to the input file (example: *requirements.in*).

4.1.2 *os-name*

Required Operating system name (platform slug).

4.1.3 *github-token*

GitHub Token for creating pull requests.

4.1.4 *output-directory*

The path to directory to place the compiled requirements (default value: *requirements/*).

4.1.5 *target-branch*

Branch to make a pull request to.

4.1.6 *prefix*

Prefix for the destination compiled filename.

4.1.7 *create-artifact*

Flag to determine if an artifact should be created (default value: *false*).

CHAPTER

FIVE

METHODOLOGY

5.1 Selectively pick workflows to run

Selectively pick which workflow to run based on what has changed.

1. Monitor your input requirement files using `paths` directive.
2. Do not trigger your main tests on changes in input requirements.
3. Trigger your main tests if compiled requirements have changed.

5.2 Use PAT instead of default GITHUB_TOKEN on private repositories

To create a Personal access token (PAT), go to GitHub Settings -> Developer Settings -> Fine-grained tokens and create a new token on your repository with the following permissions:

- `Actions`: Read and write.
- `Contents`: Read and write.
- `Environments`: Read-only.
- `Pull Requests`: Read and write.
- `Secrets`: Read-only.

Additionally, it might be useful to allow the following too:

- `Commit statuses`: Read-only.
- `Metadata`: Read-only.

Finally, create a `New repository secret` from `Secrets and variables` section of your repository `Settings`, specify `PAT_TOKEN` as `Name` and paste the content of newly created PAT as `Secret`.

5.3 Write smart Makefile commands to pick the right requirements for local installation

Makefile example

```
# Define the name of the virtual environment directory
VENV := venv

# Detect the operating system
ifeq ($(OS),Windows_NT)
    detected_OS := Windows
    PYTHON := python
    VENV_BIN := $(VENV)/Scripts
else
    detected_OS := $(shell uname)
    PYTHON := python3
    VENV_BIN := $(VENV)/bin
endif

# Define the requirement file based on the operating system
ifeq ($(detected_OS),Windows)
    REQUIREMENTS_FILE := requirements/windows-latest.txt
else ifeq ($(detected_OS),Darwin)
    REQUIREMENTS_FILE := requirements/macos-latest.txt
else ifeq ($(detected_OS),Linux)
    REQUIREMENTS_FILE := requirements/ubuntu-latest.txt
endif

# Default target
all: install

# Create a virtual environment
venv: $(VENV_BIN)/activate

# Virtual environment creation
$(VENV_BIN)/activate:
    $(PYTHON) -m venv $(VENV)

# Install requirements into the virtual environment
install: venv
    $(VENV_BIN)/pip install -r $(REQUIREMENTS_FILE)

# Enter virtual environment shell
shell: venv
    $(VENV_BIN)/python

pip-list: venv
    $(VENV_BIN)/pip list

# Clean the virtual environment
clean:
    rm -rf $(VENV)
```

(continues on next page)

(continued from previous page)

```
.PHONY: install venv clean
```

CHAPTER

SIX

EXAMPLE USAGE

.github/workflows/test-action.yml

```yaml
name: Test Compile Requirements Action

on:
  push:
    paths:
      - 'examples/requirements.in'
      - 'examples/pyproject.toml'
      - '.github/workflows/test-action.yml'
      - 'action.yml'

permissions:
  contents: write
  pull-requests: write

jobs:
  test:
    runs-on: ${{ matrix.os }}
    strategy:
      fail-fast: false
      matrix:
        os: [  # See this as an example
          ubuntu-latest,
          ubuntu-22.04,
          ubuntu-20.04,
          windows-latest,
          windows-2022,
          windows-2019,
          macos-latest,
          macos-13,
          macos-12,
          macos-11,
        ]
    steps:
      - uses: actions/checkout@v3

      - name: Set up Python 3.11
        uses: actions/setup-python@v5
        with:
```

(continues on next page)

(continued from previous page)

```
          python-version: '3.11'

      - name: Set up platform-specific variables
        id: vars
        shell: bash
        run: |
          OS_NAME=$(echo ${{ matrix.os }} | tr '[:upper:]' '[:lower:]' | sed -e 's/[^a-
↪zA-Z0-9]+/-/g')
          echo "PLATFORM_SLUG=${OS_NAME%}" >> $GITHUB_ENV
          echo "TARGET_BRANCH=$(echo ${GITHUB_REF#refs/heads/})" >> $GITHUB_ENV

      - name: Run Compile and PR Requirements Action
        uses: barseghyanartur/compile-python-requirements-action@0.1
        with:
          input-file: 'examples/requirements.in'
          os-name: ${{ env.PLATFORM_SLUG }}
          github-token: ${{ secrets.PAT_SECRET }}
          output-directory: 'examples/requirements'  # Optional
          prefix: ''  # Optional
          # Optional. Pass the target branch to the action
          target-branch: ${{ env.TARGET_BRANCH }}
          create-artifact: 'true'  # Optional

      - name: Upload Artifact
        uses: actions/upload-artifact@v3
        with:
          name: requirements-${{ env.PLATFORM_SLUG }}
          path: examples/requirements/requirements.tar.gz
          if-no-files-found: 'warn'
```

CHAPTER

SEVEN

LICENSE

MIT

CHAPTER

EIGHT

SUPPORT

For security issues contact me at the e-mail given in the *Author* section.

For overall issues, go to GitHub issues.

CHAPTER

NINE

AUTHOR

Artur Barseghyan

CHAPTER

TEN

PROJECT DOCUMENTATION

Contents:

Table of Contents

10.1 Security Policy

10.1.1 Reporting a Vulnerability

Do not report security issues on GitHub!

Please report security issues by emailing Artur Barseghyan <artur.barseghyan@gmail.com>.

10.1.2 Supported Versions

Make sure to use the latest version.

The two most recent `compile-python-requirements-action` release series receive security support.

For example, during the development cycle leading to the release of `compile-python-requirements-action` 0.17.x, support will be provided for `compile-python-requirements-action` 0.16.x.

Upon the release of `compile-python-requirements-action` 0.18.x, security support for `compile-python-requirements-action` 0.16.x will end.

Version	Supported
0.1.x	Yes

10.2 Contributor Covenant Code of Conduct

10.2.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

10.2.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks

- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

10.2.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

10.2.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

10.2.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at artur.barseghyan@gmail.com. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

10.2.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

10.2.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

10.3 Release history and notes

Sequence based identifiers are used for versioning (schema follows below):

```
major.minor[.revision]
```

- It's always safe to upgrade within the same minor version (for example, from 0.3 to 0.3.4).
- Minor version changes might be backwards incompatible. Read the release notes carefully before upgrading (for example, when upgrading from 0.3.4 to 0.4).
- All backwards incompatible changes are mentioned in this document.

10.3.1 0.1

2023-12-30

- Initial beta release.