version: 2.1

orbs:

# The python orb contains a set of prepackaged circleci configuration you can use repeatedly in your configurations files

# Orb commands and jobs help you with common scripting around a language/tool

# so you dont have to copy and paste it everywhere.

# See the orb documentation here:<https://circleci.com/developer/orbs/orb/circleci/python>

python: circleci/python@2.1.1

parameters:

eval-mode:

type: string

default: "commit"

workflows:

evaluate-commit:

when:

equal: [ commit, << pipeline.parameters.eval-mode >> ]

jobs:

- run-commit-evals:

context:

- dl-ai-courses

evaluate-release:

when:

equal: [ release, << pipeline.parameters.eval-mode >> ]

jobs:

- run-pre-release-evals:

context:

- dl-ai-courses

evaluate-all:

when:

equal: [ full, << pipeline.parameters.eval-mode >> ]

jobs:

- run-manual-evals:

context:

- dl-ai-courses

report-evals:

when:

equal: [ report, << pipeline.parameters.eval-mode >> ]

jobs:

- store-eval-artifacts:

context:

- dl-ai-courses

jobs:

run-commit-evals: # This is the name of the job, feel free to change it to better match what you're trying to do!

# These next lines defines a docker executors:<https://circleci.com/docs/2.0/executor-types/>

# You can specify an image from dockerhub or use one of the convenience images from CircleCI's Developer Hub

# A list of available CircleCI docker convenience images are available here:<https://circleci.com/developer/images/image/cimg/python>

# The executor is the environment in which the steps below will be executed - below will use a python 3.9 container

# Change the version below to your required version of python

docker:

- image: cimg/python:3.10.5

# Checkout the code as the first step. This is a dedicated CircleCI step.

# The python orb's install-packages step will install the dependencies from a Pipfile via Pipenv by default.

# Here we're making sure we use just use the system-wide pip. By default it uses the project root's requirements.txt.

# Then run your tests!

# CircleCI will report the results back to your VCS provider.

steps:

- checkout

- python/install-packages:

pkg-manager: pip

# app-dir: ~/project/package-directory/ # If your requirements.txt isn't in the root directory.

# pip-dependency-file: test-requirements.txt # if you have a different name for your requirements file, maybe one that combines your runtime and test requirements.

- run:

name: Run assistant evals.

command: python -m pytest --junitxml results.xml test\_assistant.py

- store\_test\_results:

path: results.xml

run-pre-release-evals:

docker:

- image: cimg/python:3.10.5

steps:

- checkout

- python/install-packages:

pkg-manager: pip

# app-dir: ~/project/package-directory/ # If your requirements.txt isn't in the root directory.

# pip-dependency-file: test-requirements.txt # if you have a different name for your requirements file, maybe one that combines your runtime and test requirements.

- run:

name: Run release evals.

command: python -m pytest --junitxml results.xml test\_release\_evals.py

- store\_test\_results:

path: results.xml

run-manual-evals:

docker:

- image: cimg/python:3.10.5

steps:

- checkout

- python/install-packages:

pkg-manager: pip

# app-dir: ~/project/package-directory/ # If your requirements.txt isn't in the root directory.

# pip-dependency-file: test-requirements.txt # if you have a different name for your requirements file, maybe one that combines your runtime and test requirements.

- run:

name: Run end to end evals.

command: python -m pytest --junitxml results.xml test\_assistant.py test\_release\_evals.py

- store\_test\_results:

path: results.xml

store-eval-artifacts:

docker:

- image: cimg/python:3.10.5

steps:

- checkout

- python/install-packages:

pkg-manager: pip

# app-dir: ~/project/package-directory/ # If your requirements.txt isn't in the root directory.

# pip-dependency-file: test-requirements.txt # if you have a different name for your requirements file, maybe one that combines your runtime and test requirements.

- run:

name: Save eval to html file

command: python save\_eval\_artifacts.py

- store\_artifacts:

path: /tmp/eval\_results.html

destination: eval\_results.html