

# Ascender

Accelerator of Scientific Development and Research

XCCV GROUP



cvpaper.challenge

Jul. 7th, 2022

# Content

- What is Ascender?
- Create repo from Ascender
- Directory structure
- How to use Ascender
- Technical stack of Ascender
- Upcoming feature

# What is Ascender?

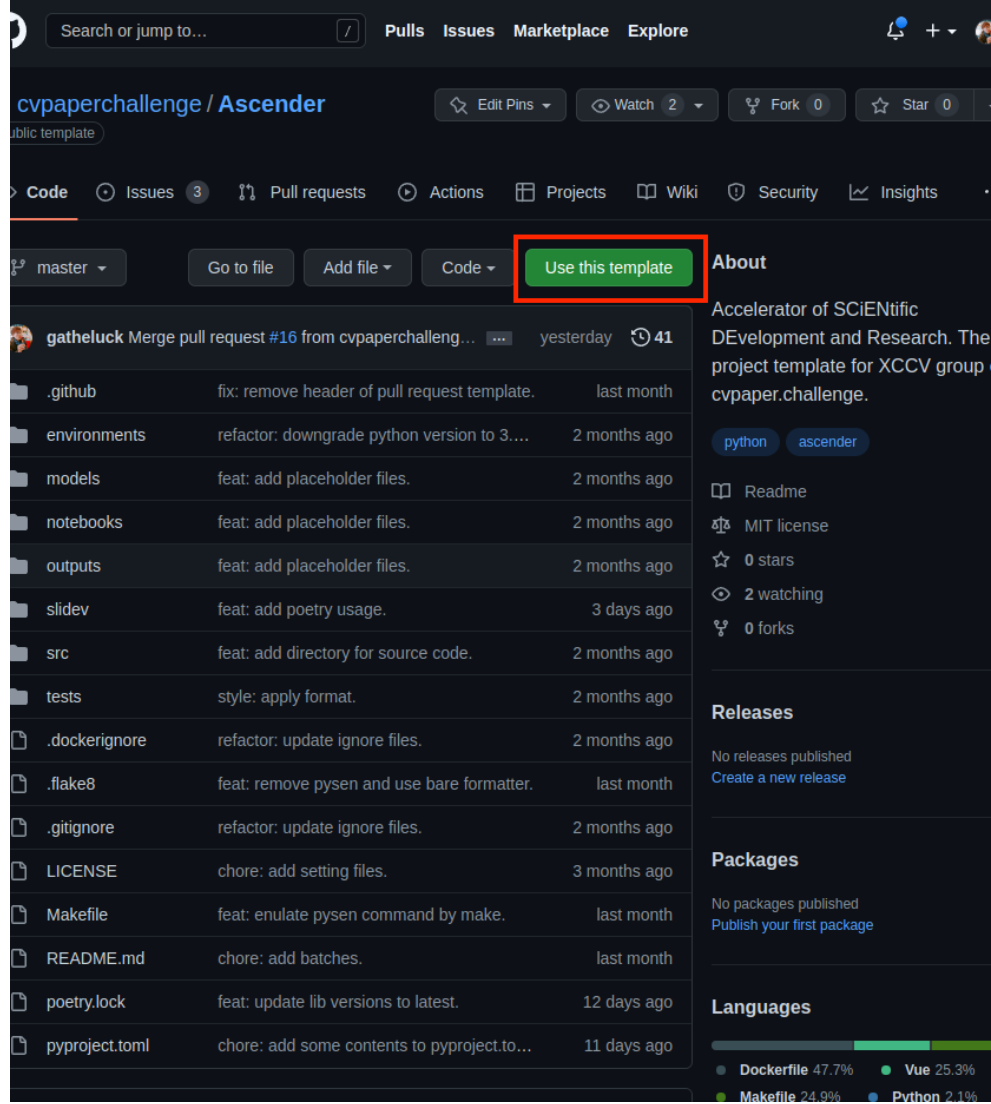
- 主にPythonを開発言語として使用した研究プロジェクト向けのGitHubのリポジトリテンプレート.
- 開発時に有用な機能を多数サポート:
  - コンテナ仮想化: Dockerを使用することで開発環境依存を減らし, コードの移植性を向上.
  - 仮想環境構築 / パッケージ管理: Poetryを使用したパッケージ管理により, 同一環境の再現性を向上.
  - コードスタイル: Black, Flake8, isortを使用してコードスタイルを統一.
  - 静的型チェック: Mypyによる静的型チェックによりコードの信頼性を向上.
  - テスト: pytestを使用したテストコードを簡単に追加可能.
  - GitHub関連機能: Pull request時のスタイル確認・テストのworkflowやissueテンプレート等を実装済.
- <https://github.com/cvpaperchallenge/Ascender> からアクセス可能.

# Create repo from Ascender

以下の手順でAscenderをテンプレートとしてGitHubリポジトリが作成可能.

- AscenderのGitHubリポジトリにアクセス.
- "Use this template"ボタンをクリック.
- 作成するリポジトリの情報を入力し,  
"Create repository from template"をクリック.

テンプレートからリポジトリを作成する方法はGitHubのドキュメントにも説明がある.



# Directory structure

``src`` ディレクトリ下にパッケージの形でコードを追加していく構成.

```
$ tree -L 1 --dirsfirst # カレントディレクトリ下のディレクトリとファイルを, 1階層の深さだけ, ディレクトリから先に表示するコマンド.
```

```
.
├─ data           # <- データセットを格納するためのディレクトリ.
├─ environments  # <- Docker関連のファイルを格納するディレクトリ.
├─ models        # <- 学習済みのMLモデル等を格納するためのディレクトリ.
├─ notebooks     # <- Jupyter Notebookを格納するためのディレクトリ.
├─ outputs       # <- コード実行時の出力を格納するためのディレクトリ.
├─ src           # <- メインの開発物を格納するディレクトリ. このディレクトリ下がPythonパッケージとなることを想定.
├─ tests         # <- pytestのテストを格納するディレクトリ. 全てのテストはこのディレクトリ下に追加.
├─ LICENSE       # <- ライセンス情報を記載するファイル.
├─ Makefile      # <- タスクランナーとして使用しているMakeの設定ファイル. 頻出のタスクを短いコマンドで実行可能.
├─ poetry.lock   # <- Poetryが自動変更するファイル. IMPORTANT: 直接編集しないで下さい!
├─ pyproject.toml # <- Projectの設定を記載するファイル. Poetry, Black, Flake8, isort, Mypy等の設定が記載.
└─ README.md     # <- Ascenderについての基本的な説明が記載されているファイル.
```

# How to use Ascender

## Start development

```
# cpu環境の場合は`cd environments/cpu`  
$ cd environments/gpu  
  
# Dockerfileからイメージをビルドし、  
# ビルドされたイメージからコンテナを作成。  
# dオプションをつけるとコンテナをバックグラウンドで実行。  
$ sudo docker compose up -d  
  
# コンテナ内でbashを実行(コンテナ内に入る)。  
$ sudo docker compose exec core bash  
  
# Poetryで仮想環境を構築し、依存パッケージをインストール。  
$ poetry install
```

- 初回の実行時は`sudo docker compose up -d`に時間がかかる。(環境によっては`docker`の実行に`sudo`が必要ない場合がある.)
- `sudo docker compose up -d`によってAscenderのルートディレクトリ以下がコンテナ内のワーキングディレクトリにマウントされる。
- `poetry install`は初回のみ実行すれば良い。(Poetryの使い方については別スライドに記載.)

# How to use Ascender

## During development

```
# Poetryを使用して新しいパッケージを仮想環境にインストール。  
$ poetry add torch
```

```
# 仮想環境内でコマンドを実行するときは`poetry run`を使用。  
# 例. 仮想環境内でPythonのインタラクティブシェルを起動。  
$ poetry run python3
```

```
# Makefile内に頻出するタスクの短縮コマンドを定義。  
# `src`と`test`下のファイルのコードスタイルを自動整形。  
$ make format
```

```
# `src`と`test`下のファイルのコードスタイルチェックと  
# 静的型チェックを行い, pytestでテストコードを実行。  
$ make test-all
```

- 開発時はコンテナ内にPoetryが作成した仮想環境を使用.  
(Poetryの使い方については別スライドに記載.)
- Makeをタスクランナーとして使用し, 頻出するタスクを  
`make` コマンドで簡単に実行可能.
- GitHubでPull Requestを作成時にスタイルチェック, 静的型チェック, テストコード実行を行うCI用のworkflowも実装済.

# How to use Ascender

## Stop development



```
# cpu環境の場合は`cd environments/cpu`  
$ cd environments/gpu  
  
# コンテナを停止。  
$ sudo dokcer compose stop  
  
# もしコンテナを削除したい場合は停止と削除を一度に行うことも可能。  
$ sudo docker compose down
```

- 基本的にコンテナをバックグラウンドで実行し続けていても問題無いが、コンテナを停止したいときは`docker compose stop`を使用. 再度立ち上げるときは"Start development"のページの処理を繰り返せば良い.
- `docker compose down`を使用するとコンテナを削除可能.





# Technical stack of Ascender

## Basic knowledge is required

-  Poetry: Pythonパッケージの依存関係管理, パッケージ作成を行うためのツール.
  - 基本的な使い方は別スライド"[Poetry101](#)" (En, Ja) を参照.
-  Docker: コンテナ仮想化を使用した開発を行うためのツール.
  - よく分からなくても, Ascenderの`README.md`と本スライドに記載の手順に沿うだけで基本的な開発は可能.

## Better to know

-  Mypy: Pythonで静的型チェックを行うためのツール.
-  pytest: Pythonでテストを簡単に書くためのツール.

※ Poetry以外のツールの基本的な使い方のスライドも順次作成予定.

# Upcoming feature

## Project page template

- カッコよくて, 見やすいプロジェクトページを短時間で作りたいが, フロントエンド開発の学習にかけたくない.
- SPA形式のテンプレート:
  - 頻出する機能をコンポーネントとして実装済.
  - HTML + CSS の知識のみで編集可能. (コンポーネントによってはJavaScriptも必要.)
  - マルチデバイス対応済.
- 2022年7月末に追加予定.

# NeRF

## Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020 Oral - Best Paper Honorable Mention

Ben Mildenhall<sup>1\*</sup>  
Jonathan T. Barron<sup>2</sup>

<sup>1</sup>UC Berkeley

Pratul P. Srinivasan<sup>1\*</sup>  
Ravi Ramamoorthi<sup>3</sup>

<sup>2</sup>Google Research

Matthew Tancik<sup>1\*</sup>  
Ren Ng<sup>1</sup>

<sup>3</sup>UC San Diego

\*Denotes Equal Contribution



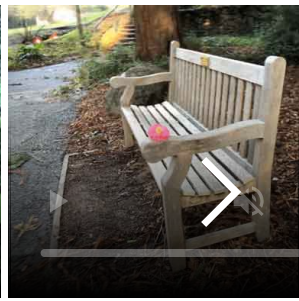
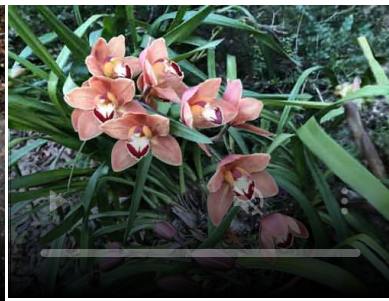
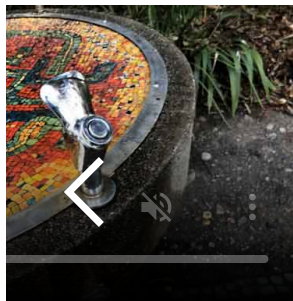
Paper

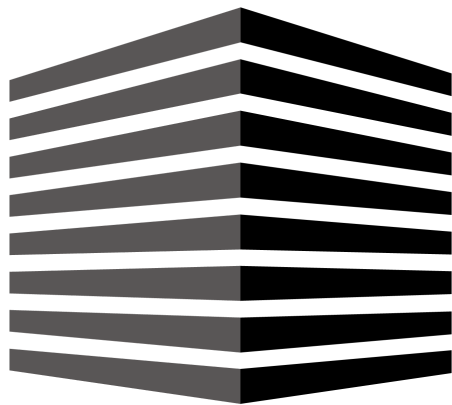


</Code>



Data





**cvpaper.challenge**

Visit our page

END