Awesome Streamlit Documentation

Marc Skov Madsen

Jul 28, 2020

Awesome Streamlit Docs!

This is the documentation of the **Awesome Streamlit Project** including the

- Awesome Streamlit Resources List on GitHub
- Repo on GitHub
- App at awesome-streamlit.org
- Docs on Read The Docs
- Python Package on PyPi
- Docker Image on Docker Hub

CHAPTER 1

Awesome Streamlit Resources Awesome

1.1 Alternative

- Bokeh (#Alternative)
- Jupyter Voila (#Alternative)
- Panel (#Alternative)
- Plotly Dash (#Alternative)

1.2 App

- Hello-streamlit deployed on Glitch by Alexander Garcia (#App, #Deployment)
- How to create and deploy data exploration web app easily using python (#App, #Code, #Deployment)
- Kaggle Mushrooms Dashboard (#App)
- NLP Based App with Streamlit by Jesse E. Agbe (JCharis) (#App, #Code)
- Sentiment Analyzer Tool (#App, #Code, #Social)
- Streamlit Demo by Luke Posey (#App, #Code)
- Streamlit-components-demo App (#App)

1.3 Awesome-Streamlit.org

- App by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- Docker Image by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- Docs by Awesome-Streamlit.org (#Awesome-Streamlit.org)

- LinkedIn post that started awesome-streamlit.org (#Awesome-Streamlit.org, #Social)
- Python Package by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- Repo by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- Resources List by Awesome-Streamlit.org (#Awesome-Streamlit.org)

1.4 Code

- Awesome Streamlit Test Runner by Marc Skov Madsen (#App In Gallery, #Code)
- Country Indicators by Marc Skov Madsen (#App In Gallery, #Code, #Voila)
- Deploying Streamlit app to EC2 instance (#Code)
- Iris EDA App by Jesse E. Agbe (JCharis) (#App In Gallery, #Code)
- NBA Roster Turnover by Kevin Arvai (#App In Gallery, #Code)
- Self Driving Cars by Streamlit (#App In Gallery, #Code)
- Sentiment Algorithm by Paras Patidar (#App In Gallery, #Code, #Machine Learning)
- SpacyIO by Ines Montani (#App In Gallery, #Code, #NLP)
- Spreadsheet by Marc Skov Madsen (#App In Gallery, #Code)
- Streamlit-components-demo Code (#Code)
- Uber Data Explorer App (#Code)
- Uber NYC Pickups by Streamlit (#App In Gallery, #Code)
- Yahoo Finance by Marc Skov Madsen (#App In Gallery, #Code, #Finance)

1.5 Social

- LinkedIn #streamlit (#Social)
- Twitter #streamlit (#Social)

1.6 Streamlit.io

- Streamlit Community (#Streamlit.io)
- Streamlit Docs (#Streamlit.io)
- Streamlit launches article in TechCrunch (#Streamlit.io)
- Streamlit.io (#Streamlit.io)
- The announcing blog (#Streamlit.io)
- The announcing community post (#Streamlit.io)

1.7 Technical

- Hacker News technical discussion of how Streamlit work (#Technical)

1.8 Tutorial

- A step by step guide to running streamlit pytorch and bert on a cheap aws instance (#Article, #Deployment, #Tutorial)
- Building a ui for your latest ai by Luke Posey (#Article, #Tutorial)
- Building an Iris EDA App with Streamlit and Python by Jesse E. Agbe (JCharis) (#Tutorial, #Video)
- Full-Stack AI: Building a UI for Your Latest AI Project in No Time at All (#Article, #Image Recognition, #Tutorial)
- How To Deploy Streamlit Apps (Using Heroku) (#Deployment, #Tutorial, #Video)
- How to build your machine learning app in 3 simple steps (#Article, #Deployment, #Tutorial)
- How to create and deploy data exploration web app easily using python (#Article, #Deployment, #Tutorial)
- How to write web apps using simple python for data scientists (#Article, #Tutorial)
- Streamlit Python Tutorial (Crash Course) by Jesse E. Agbe (JCharis) (#Tutorial, #Video)

CHAPTER 2

Awesome Streamlit Awesome

The fastest way to build **Awesome Tools and Apps**! Powered by **Python**!

The purpose of this project is to share knowledge on how Awesome Streamlit is and can become. Pull requests are very welcome!

Streamlit has just been announced (Oct 2019) but I see the potential of becoming the **Iphone of Data Science Apps**. And maybe it can even become the Iphone of Technical Writing, Code, Micro Apps and Python.

This project provides

- A curated list of Awesome Streamlit **resources**. See below.
- An **awesome Streamlit application** with a **gallery** of Awesome Streamlit Apps.
  - Feel free to add your awesome app to the gallery via a Pull request. It's easy (see below).
- A **vision** on how awesome Streamlit is and can become.
- A **best practices** example and **starter template** of an awesome, multipage app with an automated CI/ CD pipeline, deployed to the cloud and running in a Docker container.

Visit the app at awesome-streamlit.org!

Awesome Streamlit Org Animation

2.1 The Magic of Streamlit

The only way to truly understand how magical Streamlit is to play around with it. But if you need to be convinced first, then here is the **4 minute introduction** to Streamlit!

Afterwards you can go to the Streamlit docs to get started. You might also visit Awesome Streamlit docs.

Introduction to Streamlit

2.2 Awesome Resources

A curated list of awesome streamlit resources. Inspired by awesome-python and awesome-pandas.

2.2.1 Alternative

- Bokeh (#Alternative)
- Jupyter Voila (#Alternative)
- Panel (#Alternative)
- Plotly Dash (#Alternative)

2.2.2 App

- Hello-streamlit deployed on Glitch by Alexander Garcia (#App, #Deployment)
- How to create and deploy data exploration web app easily using python (#App, #Code, #Deployment)
- Kaggle Mushrooms Dashboard (#App)
- NLP Based App with Streamlit by Jesse E. Agbe (JCharis) (#App, #Code)
- Powering up Apache JMeter with Streamlit by Naveen Kumar (#App, #Code)
- Quickly Build and Deploy a Dashboard with Streamlit by Maarten Grootendorst (#App, #Code, #Deployment)
- Sentiment Analyzer Tool (#App, #Code, #Social)
- Streamlit Demo by Luke Posey (#App, #Code)
- Streamlit-components-demo App (#App)

2.2.3 Article

- Intermediate Streamlit - Tips and Tricks for an evolving app by Peter Baumgartner (#Article)

2.2.4 Awesome-Streamlit.org

- App by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- Docker Image by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- Docs by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- LinkedIn post that started awesome-streamlit.org (#Awesome-Streamlit.org, #Social)
- Python Package by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- Repo by Awesome-Streamlit.org (#Awesome-Streamlit.org)
- Resources List by Awesome-Streamlit.org (#Awesome-Streamlit.org)

2.2.5 Code

- Awesome Streamlit Test Runner by Marc Skov Madsen (#App In Gallery, #Code)
- Bokeh Experiments by Marc Skov Madsen (#App In Gallery, #Code)
- Country Indicators by Marc Skov Madsen (#App In Gallery, #Code, #Voila)
- Deploying Streamlit app to EC2 instance (#Code)
- Emojis by Marc Skov Madsen (#App In Gallery, #Code)
- File Download Workaround by Marc Skov Madsen (#App In Gallery, #Code)
- File Uploader by Marc Skov Madsen (#App In Gallery, #Code)
- File Uploader - Multiple Files by Marc Skov Madsen (#App In Gallery, #Code)
- Image Classifier by Marc Skov Madsen (#App In Gallery, #Code)
- Iris Classifier by Noah Saunders (#App In Gallery, #Code, #Machine Learning)
- Iris EDA App by Jesse E. Agbe (JCharis) (#App In Gallery, #Code)
- Kickstarter Dashboard by Marc Skov Madsen (#App In Gallery, #Code)
- Layout Experiments by Marc Skov Madsen (#App In Gallery, #Code)
- ML App registry by Boadzie Daniel (#App In Gallery, #Code, #Machine Learning)
- Medical Language Learner Model by Georgi Tancev (#App In Gallery, #Code, #NLP)
- Mining and Classifying Medical Text Documents by Georgi Tancev (#App, #Code)
- NBA Roster Turnover by Kevin Arvai (#App In Gallery, #Code)
- Owid Dashboard by Marc Skov Madsen (#App In Gallery, #Code)
- Self Driving Cars by Streamlit (#App In Gallery, #Code)
- Sentiment Algorithm by Paras Patidar (#App In Gallery, #Code, #Machine Learning)
- SpacyIO by Ines Montani (#App In Gallery, #Code, #NLP)
- Spreadsheet by Marc Skov Madsen (#App In Gallery, #Code)
- Streamlit-components-demo Code (#Code)
- Table Experiments by Marc Skov Madsen (#App In Gallery, #Code)
- Uber Data Explorer App (#Code)
- Uber NYC Pickups by Streamlit (#App In Gallery, #Code)
- Yahoo Finance by Paduel (#App In Gallery, #Code, #Finance)

2.2.6 Guide

- Host Streamlit on Heroku with Nginx basic authentication by Alexandre Domingues (#Code, #Guide)

2.2.7 Sister Sites

- Awesome-Panel.Org by Marc Skov Madsen (#Sister Sites)

2.2.8 Social

- LinkedIn #streamlit (#Social)
- Twitter #streamlit (#Social)

2.2.9 Streamlit.io

- Streamlit Community (#Streamlit.io)
- Streamlit Docs (#Streamlit.io)
- Streamlit launches article in TechCrunch (#Streamlit.io)
- Streamlit.io (#Streamlit.io)
- The announcing blog (#Streamlit.io)
- The announcing community post (#Streamlit.io)

2.2.10 Technical

- Hacker News technical discussion of how Streamlit work (#Technical)

2.2.11 Tutorial

- A step by step guide to running streamlit pytorch and bert on a cheap aws instance (#Article, #Deployment, #Tutorial)
- Building a ui for your latest ai by Luke Posey (#Article, #Tutorial)
- Building an Iris EDA App with Streamlit and Python by Jesse E. Agbe (JCharis) (#Tutorial, #Video)
- Full-Stack AI: Building a UI for Your Latest AI Project in No Time at All (#Article, #Image Recognition, #Tutorial)
- How To Deploy Streamlit Apps (Using Heroku) (#Deployment, #Tutorial, #Video)
- How to build your machine learning app in 3 simple steps (#Article, #Deployment, #Tutorial)
- How to create and deploy data exploration web app easily using python (#Article, #Deployment, #Tutorial)
- How to write web apps using simple python for data scientists (#Article, #Tutorial)
- Mining and Classifying Medical Text Documents by Georgi Tancev (#Article, #Deployment, #NLP, #Tutorial)
- Streamlit Python Tutorial (Crash Course) by Jesse E. Agbe (JCharis) (#Tutorial, #Video

2.3 Governance

This repo is maintained by me :-)

I'm Marc, Skov, Madsen, PhD, CFA®, Lead Data Scientist Developer at Ørsted

You can learn more about me at datamodelsanalytics.com

I try my best to govern and maintain this project in the spirit of the Zen of Python.

But **i'm not an experienced open source maintainer** so helpfull suggestions are appreciated.

Thanks

2.4 Contribute

GitHub Issues and Pull requests are very welcome!

If you believe Awesome Streamlit is awesome and would like to join as a Core Developer feel free to reach out via datamodelsanalytics.com

2.4.1 How to contribute awesome links

The best way to contribute an awesome link is via a Pull request.

In the pull request you should

- describe why your contribution is awesome and should be included.
- add your resource the list of RESOURCES in the package/awesome_streamlit/database/resources.py file.

Thanks.

2.4.2 How to contribute awesome apps

The best way to contribute an awesome app is via a Pull request.

In the pull request you should

- describe why your contribution is awesome and should be included.
- create a new folder `gallery/<your_app_name>` and app file `gallery/<your_app_name>/<your_app_name.py>`.
- Add your app code conforming to the template

```
"""
## APP NAME

DESCRIPTION

Author: [YOUR NAME](https://URL_TO_YOU))\n
Source: [Github](https://github.com/URL_TO_CODE)
"""
import streamlit as st

# Your imports goes below

def main():
    st.title("APP NAME")
    st.markdown("DESCRIPTION")

    # Your code goes below

if __name__ == "__main__":
    main()
```

- Please note magic in sub pages does not work. So **don't use magic**.

- add the `your_app_name` to the
  - list of APPS_IN_GALLERY in the package\awesome_streamlit\database\apps_in_gallery.py.
- update the requirements_base.txt file. Please specify the required versions.
- Run the automated tests using `invoke test.all` and fix all errors from your app
- Run the full app via `streamlit run app.py` and manually test your contribution.

Please note that your app should not require high compute power as we are running on one of the cheapest tiers available on Azure.

Feel free to reach out if you have comments, questions or need help.

Thanks.

2.4.3 How to contribute to the Streamlit Community

Please sign up to and participate in the community at discuss.streamlit.io

2.4.4 How to contribute to the Streamlit Package

Please contribute to improving the Streamlit package at GitHub/streamlit/streamlit

2.4.5 How to contribute to Streamlit.io

Streamlit.io is in the position of trying to balance building an awesome, succesfull business and providing an awesome product to the open source community.

If you are in a Team please consider signing up for the beta of

- Streamlit for teams

2.4.6 How to sponsor the Awesome Streamlit project

If you would like to sponsor my time or the infrastructure the platform is running on, feel free to reach out via datamodelsanalytics.com.

You can also appreciate the work I have already done if you

Buy me a coffee

Thanks

Marc

2.5 LICENSE

Attribution-ShareAlike 4.0 International

2.6 Getting Started with the Awesome Streamlit Repository

2.6.1 Prerequisites

- An Operating System like Windows, OsX or Linux
- A working Python installation.
  - We recommend using 64bit Python 3.7.4.
- a Shell
  - We recommend Git Bash for Windows 8.1
  - We recommend wsl for For Windows 10
- an Editor
  - We recommend VS Code (Preferred) or PyCharm.
- The Git cli

2.6.2 Installation

Clone the repo

```
git clone https://github.com/MarcSkovMadsen/awesome-streamlit.git
```

cd into the project root folder

```
cd awesome-streamlit
```

Create virtual environment

via python

Then you should create a virtual environment named .venv

```
python -m venv .venv
```

and activate the environment.

On Linux, OsX or in a Windows Git Bash terminal it's

```
source .venv/Scripts/activate
```

or alternatively

```
source .venv/bin/activate
```

In a Windows terminal it's

```
.venv/Scripts/activate.bat
```

or via anaconda

Create virtual environment named awesome-streamlit

```
conda create -n awesome-streamlit python=3.7.4
```

and activate environment.

```
activate awesome-streamlit
```

Then you should install the local requirements

```
pip install -r requirements_local.txt
```

Finally you need to install some spacy dependencies

```
python -m spacy download en_core_web_sm
python -m spacy download en_core_web_md
python -m spacy download de_core_news_sm
```

2.6.3 Build and run the Application Locally

```
streamlit run app.py
```

or as a Docker container via

```
invoke docker.build --rebuild
invoke docker.run-server
```

2.6.4 Run the Application using the image on Dockerhub

If you don’t wan’t to clone the repo and build the docker container you can just use `docker run` to run the image from Dockerhub

To run bash interactively

```
docker run -it -p 80:80 --entrypoint "/bin/bash" marcskovmadsen/awesome-
↪streamlit:latest
```

To run the streamlit interactively on port 80

```
docker run -it -p 80:80 --entrypoint "streamlit" marcskovmadsen/awesome-
↪streamlit:latest run app.py
```

2.6.5 Code quality and Tests

We use

- isort for sorting import statements
- autoflake to remove unused imports and unused variables
- black the opinionated code formatter

- pylint for static analysis
- mypy for static type checking
- pytest for unit to functional tests

to ensure a high quality of our code and application.

You can run all tests using

```
invoke test.all
```

2.6.6 Streamlit Tests

I've created a first version of an awesome streamlit test runner. You run it via

```
streamlit run test_runner_app.py
```

or in Docker

```
docker run -it -p 80:80 --entrypoint "streamlit" marcskovmadsen/awesome-
↪streamlit:latest run test_runner_app.py
```

Awesome Streamlit Test Runner

2.6.7 Workflow

We use the power of Invoke to semi-automate the local workflow. You can see the list of available commands using

```
$ invoke --list
Available tasks:

  docker.build                            Build Docker image
  docker.push                             Push the Docker container
  docker.run                              Run the Docker container interactively.
  docker.run-server                       Run the Docker container interactively
  docker.system-prune                     The docker system prune command will free␣
↪up space
  test.all (test.pre-commit, test.test)   Runs isort, autoflake, black, pylint, mypy␣
↪and pytest
  test.autoflake                          Runs autoflake to remove unused imports on␣
↪all .py files recursively
  test.bandit                             Runs Bandit the security linter from PyCQA.
  test.black                              Runs black (autoformatter) on all .py files␣
↪recursively
  test.isort                              Runs isort (import sorter) on all .py files␣
↪recursively
  test.mypy                               Runs mypy (static type checker) on all .py␣
↪files recursively
  test.pylint                             Runs pylint (linter) on all .py files␣
↪recursively to identify coding errors
  test.pytest                             Runs pytest to identify failing tests
```

2.6.8 Configuration

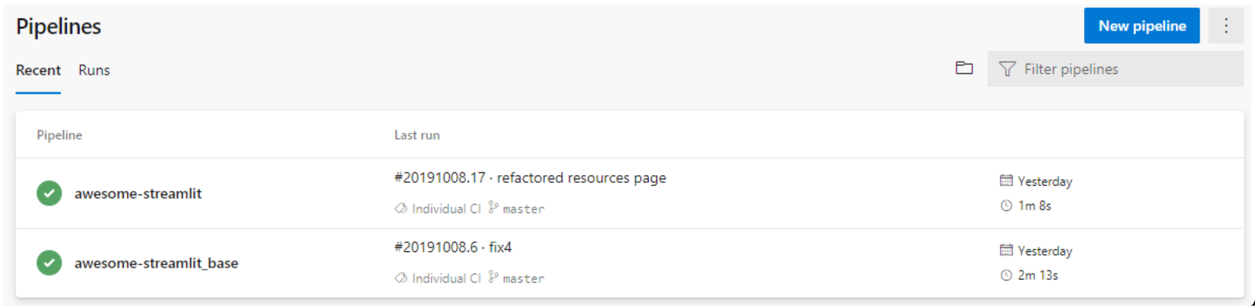
You can configure the app in the `config.py` file.

Please note that Streamlit has its own config files in the `~/.streamlit` folder.

2.6.9 CI/ CD and Hosting

The application is

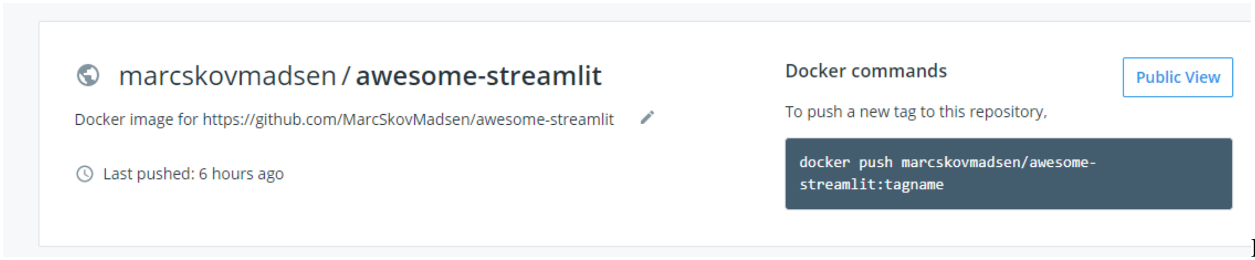
- build as a Docker image and tested via Azure Pipelines builds
  - You find the Dockerfiles here and the Azure pipelines yml files here



Pipeline	Last run	
awesome-streamlit	#20191008.17 · refactored resources page Individual CI master	Yesterday 1m 8s
awesome-streamlit_base	#20191008.6 · fix4 Individual CI master	Yesterday 2m 13s

Azure Pipelines

- pushed to the Dockerhub repository marcskovmadsen/awesome-streamlit.



Docker image for https://github.com/MarcSkovMadsen/awesome-streamlit

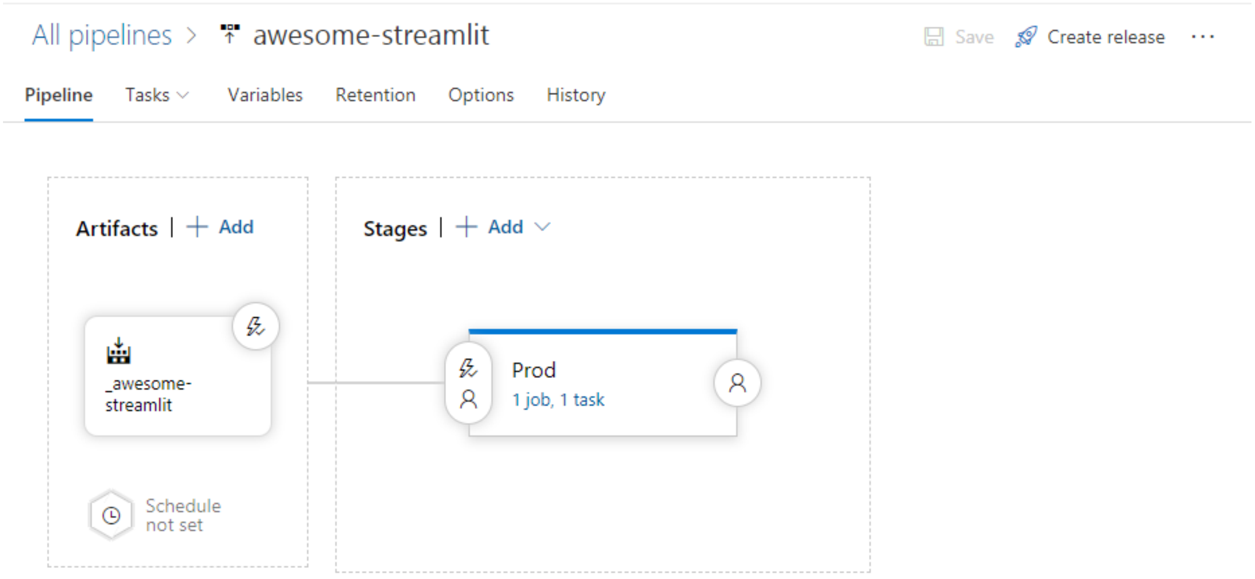
Last pushed: 6 hours ago

To push a new tag to this repository,

```
docker push marcskovmadsen/awesome-streamlit:tagname
```

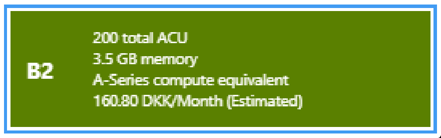
Dockerhub

- released via Azure Pipelines



Azure Pipelines

- to a web app for containers service on Azure on the cheapest non-free pricing tier


B2
200 total ACU
3.5 GB memory
A-Series compute equivalent
160.80 DKK/Month (Estimated)

Azure Pipelines

2.6.10 The Awesome-Streamlit Package

You can build the package using

```
cd package
python setup.py sdist bdist_wheel
```

If you wan't to publish the package to PyPi you should first

update the version number in the setup.py file. The format is `YYYYmmdd.version`. For example `20191014.2`

Then you run

```
twine upload dist/awesome-streamlit-YYYYmmdd.version.tar.gz -u <the-pypi-username> -p
↪<the-pypi-password>
```

For more info see the package README.md

2.6.11 Project Layout

The basic layout of a application is as simple as

```
.
└── app.py
```

As our application grows we would refactor our app.py file into multiple folders and files.

- *assets* here we keep our css and images assets.
- *models* - Defines the layout of our data in the form of
  - Classes: Name, attribute names, types
  - DataFrame Schemas: column and index names, dtypes
  - SQLAlchemy Tables: columns names, types
- *pages* - Defines the different pages of the Streamlit app
- *services* - Organizes and shares business logic, models, data and functions with different pages of the Streamlit App.
  - Database interactions: Select, Insert, Update, Delete
  - REST API interactions, get, post, put, delete
  - Pandas transformations

and end up with a project structure like

```
.
├── app.py
└── src
    └── assets
```

(continues on next page)

(continued from previous page)

```
    |     └── css
    |     |    ├── app.css
    |     |    ├── component1.css
    |     |    ├── component2.css
    |     |    ├── page1.css
    |     |    └── page2.css
    |     └── images
    |     |    ├── image1.png
    |     |    └── image2.png
    ├── core
    |   └── services
    |        ├── service1.py
    |        └── service2.py
    └── pages
    |   └── pages
    |        ├── page1.py
    |        └── page2.py
    └── shared
        └── models
        |    ├── model1.py
        |    └── model2.py
        └── components
             ├── component1.py
             └── component2.py
```

Further refactoring is guided by by this blog post and the Angular Style Guide.

We place our tests in a `test` folder in the root folder organized with folders similar to the `app` folder and file names with a `test_` prefix.

```
.
└── test
    ├── test_app.py
    ├── core
    |   └── services
    |        ├── test_service1.py
    |        └── test_service2.py
    └── pages
    |   └── pages
    |        ├── page1
    |        |   └── test_page1.py
    |        └── page2
    └── shared
        └── models
        |    ├── test_model1.py
        |    └── test_model2.py
        └── components
             ├── test_component1.py
             └── test_component2.py
```

CHAPTER 3

Awesome Streamlit Awesome

3.1 Disclaimer

THE BELOW IS VERY PRELIMINARY AND RAPIDLY DEVELOPING! IT'S A VISION and a HYPOTHESIS. NOT FACTS! MIGHT CHANGE A LOT.

I JUST DISCOVERED JUPYTER VOILA WHICH MIGHT BE AN ALTERNATIVE. I DO NOT KNOW HOW THAT COMPARES. I WILL DESCRIBE THAT BELOW ASAP.

3.2 Introduction

This article will discuss if Streamlit is awesome and how awesome it can be.

Streamlit is announced as being **The fastest way to build custom ML tool** but we believe it has the potential to become **much more awesome** than that.

Streamlit Video

The best way to think about the arrival of Streamlit is to think of the **arrival of the Iphone** being announced as **the smartest way to place calls and send text messages**. Well yes :-). But now we can see that it's a powerfull compute engine and an ecosystem of apps that has changed peoples lifes.

Streamlit has just arrived (Oct 2019). So it's too early tell if Streamlit will become truly awesome.

But we see the **potential to become the Iphone of Data Science, Technical Writing, Micro Apps, Code, Python and more**. It has the potential to change the way we share knowledge and create tools and applications because it's so plain simple to use.

This article was inspired by a reply in this post on LinkedIn and some replies to the announcement in the Streamlit Community.

You can find a curated list of Awesome Streamlit resources here

3.3 Contribute

Feel free to contribute your comments or suggestions to to this article.

You can contribute through GitHub Issues or Pull requests.

3.4 Data Science

Data Science is a very broad category. It's used for to categorize work such as

- data transformations and visualisations in excel
- interactive data visualisations in BI Tools like Power BI and Tableau
- Building Fundamental, Statistical, Mathematical, Machine Learning or Deeplearning models in Python, R etc.
- Building applications on top of data science models and deploying them to production using web frameworks such as Bokeh/ Dash, Flask/ Django and React/Vue/Angular.

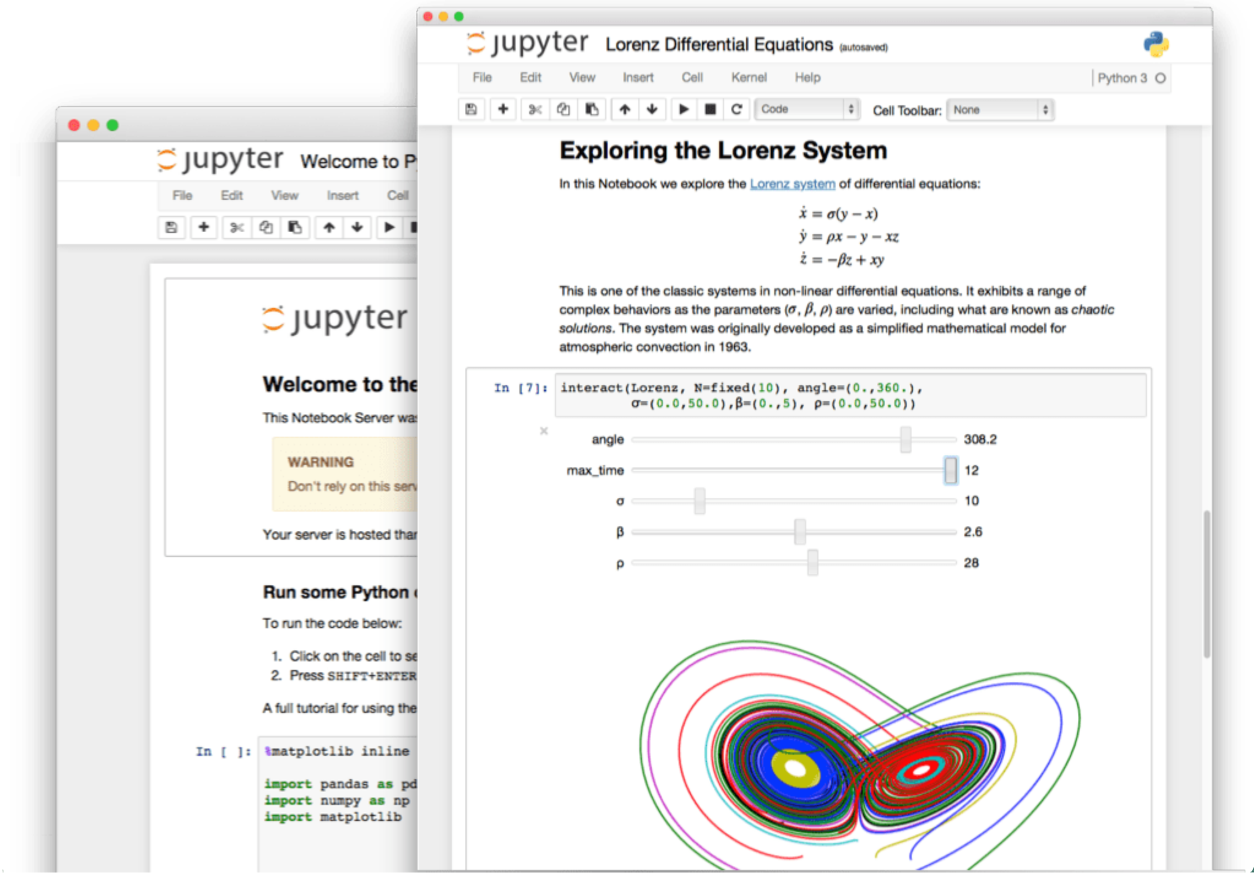
and much more

But actually every body extracting, transforming, loading and presenting data are actually doing work very similar to data scientists. If they are very structured about it and using methodologies from software development they are called Data Engineers, ML Engineers or Platform Engineers :-)

3.4.1 Notebooks

Notebooks - Situation

One of the revolutionary tools of modern data science is the Jupyter Notebook.


jupyter Welcome to P
File Edit View Insert Cell
jupyter
jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None

Welcome to the

This Notebook Server wa

WARNING
Don't rely on this serv

Your server is hosted than

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

```
   angle  308.2
max_time  12
       σ  10
       β  2.6
       ρ  28
```

Awesome

The jupyter notebook provides an environment for working very exploratory and visually within data science

Notebooks - Problem

A jupyter notebook cannot really be deployed directly to production and the users.

So Notebooks do not facilitate rapid sharing, exploration, testing and improvement cycle you would like.

Bottom line. **Notebooks makes deployment of data science products costly**.

Notebooks - Complication

The pains of the Jupyter Notebook are

- The Notebook is not as productive a development environment as an Editor or IDE.
- The Notebook workflow tends to lead to code that is poorly written and/ or is difficult to reproduce.
- The Notebook cannot really be deployed to production and shared with users as an application. You can distribute it as pdf or html. But most often it will be refactored into a code library or an application by a team of developers.
- for more see this video by Guss Allen.

Jupyter has tried to develop a more productive development environment called Jupyter Labs.

But this environment also has it's pains.

- It's still not as productive or extensible as a real editor like VS Code or Pycharm.

- It changes the Notebook from being a simple interactive document that you can read and distribute as pdf or html to an advanced but complicated development environment that you work in.

Notebooks even more complicated

Jupyter recently (Jun 2019) announced Voila which can transform Jupyter notebooks to standalone applications and dashboards.

The approach is different than that of Streamlit. I have not yet tried Voila.

But for now I believe that the code in your editor, use caching and only transfer incremental data of Streamlit is a very strong and unique approach. Also the magical api of Streamlit is different but very, very simple to use.

Whether Streamlit can compete with the massive popularity and library of widgets of Jupyter is yet to be seen. Or whether Voila will quickly learn from Streamlit.

See Hacker News Discussion for a little bit of technical discussion.

More to come

Notebooks - Solution

Streamlit aims to solve all of the problems described above as described in the Announcement.

Streamlit Video

As you can see on the image above it simplifies everything by enabling data scientist to

- develop applications in just one python file (left side of image)
- using their editor of choice (VS Code on left side of image)
- producing a web application that can very easily be deployed to production. (right side of image)

And the application can provide the interactivity that you would normally be developing using a modern web framework as React, Vue or Angular. So **a single data scientist can develop a data science application in hours**. **No large project or team required**.

Streamlit provides **self service data science**.

And it's **very appealing to data scientists** as they can develop an application in the way they find simple, productive and fun

- Text as MarkDown
- Code in Python
- All of your data science modelling libraries at hand like Pandas, SciKitLearn, PyTorch, Keras, TensorFlow etc.
- All of your data science visualisation libraries at hand like Matplotlib, Vega and Plotly.
- No HTML and No Javascript is required.

I believe that the **Jupyter Notebook is a cell phone** and **Streamlit is the Iphone** of Data Science

3.5 Notebooks - Implications

For building data science applications

- business should start investigating Streamlit and Jupyter Voila

- There will be a lot less demand for front end developers and development in React, Vue an Angular in data science.
  - The remaining front end developers in data science should be developing specialized Web Components for Streamlit when needed.
- There will be less demand for back end developers to develop REST APIs etc. for data science products.
- Streamlit is so simple to use that business users across an enterprise can create apps them selves. It's the democratization of Data Science Apps.
  - Actually we believe the development cycle will be
    * Business users and data scientists develop data science apps independently or together and deploy to production.
    * The most valuable apps will be identified and the quality and governance of these apps will be improved and maintained. They will get an approval as Enterprise Ready apps. But probably as this is so simple and there is revision control and pull requests Business Users and Data Scientist will be able to continue the development of these apps.
- Business should start experimenting with this technology to find it's use cases and limits and evaluate how and when to start using Streamlit securely.
  - As **time to market** is so important you should **start moving!**.
- IT departments should
  - Describe their way of securely deploying these applications to on-premise or cloud and
  - join the one-click deployment solution for Teams beta.

For building larger, traditional applications with data science app components the data science app components can in a lot of cases be build in Streamlit and embedded in the larger application. REST API endpoints should still be developed when the need is there.

3.5.1 Notebooks - Disclaimer

Streamlit is very new. There will be rough edges and things you cannot yet do. And maybe it turns out the product is not secure or something else. Who knows? But as the principles and api of Streamlit is so simple and productive we believe they will be developed and fixed very rapidly.

Things you cannot do yet

- Select from thousands of web components to make your application interactive.
  - The basic ones are there
- Deploy to production with one click.
  - It can be deployed as any other python web application. And it's only one file, the streamlit package and any data science packages needed. So it's very simple to deploy one you have done it once.

More to come

3.5.2 Spreadsheets

As Streamlit apps are so simple, the quality so high and robust and it enjoys the power of Python a lot business users will transform from developing spreadsheets into developing Streamlit application.

Spreadsheets - Implications

- Train your business users in Python and Streamlit.
- Help your business users install Python and Python packages.
- Help your business users govern their projects and how to use revision control.

I believe a Spreadsheet wrapper for Streamlit removing these pain points will be provided soon as it will be so easy to setup on top of Streamlit.

3.5.3 BI Visualization Tools

Another category of revolutionary tools for data science are data reporting and visualization tools like Power BI and Tableau.

3.5.4 BI Visualization Tools - Problem

- Difficult or impossible to do model visualization.
- Hard to maintain using good software developing techniques because you do not develop code. - Does not enjoy the power of Python.

But it's still a very strong drag and drop 1-click deployment tool for data visualization.

3.5.5 Bokeh

3.5.6 Dash

Problem: You still need to master HTML and manage callbacks. Focus on Plotly charts mainly. The development cycle is slow. The api is not "magical".

Implications:

For some use case where you wan't a high degree of flexibility in layout and formatting, you might still need Dash.

3.5.7 Flask and Django

Problem: So many layers of abstraction to master. The api is not "magical".

3.5.8 React, Vue and Angular

Implications:

3.5.9 Binder

Not needed?

3.6 Technical Writing

Something like this and this will be so easy to develop in Streamlit.

3.7 Code

Streamlit lowers the barrier to entry for young and new programmers. One file in an editor is all that is needed to develop a powerfull app. Somebody will abstract away the rest in a cloud solution.

3.8 Micro Apps

When you can build something like this is a few hours you know something will change.

Video

You can find the code here. See the original tweet here

3.9 Python

Streamlit reduces the barrier to entry and enables even more people to enjoy the power of Python.

3.10 More

To come

CHAPTER 4

Contribution Guidelines

Please note that this project is released with a Contributor Code of Conduct. By participating in this project you agree to abide by its terms.

** The pull request should have a useful title. Pull requests with `Update readme.md` as title will be closed right away because I'm so tired of nobody reading this document. Please carefully read everything in `Adding to this list`.**

4.1 Table of Contents

4.2 Adding to this list

Please ensure your pull request adheres to the following guidelines:

- Search previous suggestions before making a new one, as yours may be a duplicate.
- Make sure the list is useful before submitting. That implies it has enough content and every item has a good succinct description.
- Make an individual pull request for each suggestion.
- Use title-casing (AP style).
- Use the following format: `[List Name](link)`
- Link additions should be added to the bottom of the relevant category.

- New categories or improvements to the existing categorization are welcome.
- Check your spelling and grammar.
- Make sure your text editor is set to remove trailing whitespace.
- The pull request and commit should have a useful title.
- The body of your commit message should contain a link to the repository.

Thank you for your suggestions!

4.3 Creating your own awesome list

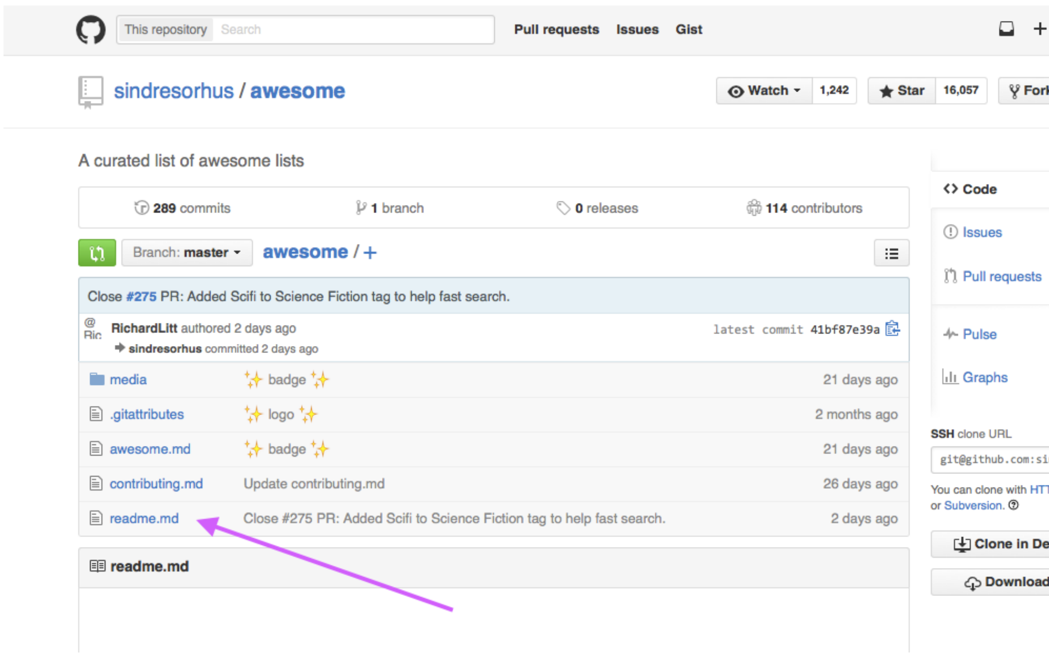
To create your own list, check out the instructions.

4.4 Adding something to an awesome list

If you have something awesome to contribute to an awesome list, this is how you do it.

You'll need a GitHub account!

1. Access the awesome list's GitHub page. For example: https://github.com/sindresorhus/awesome



A curated list of awesome lists

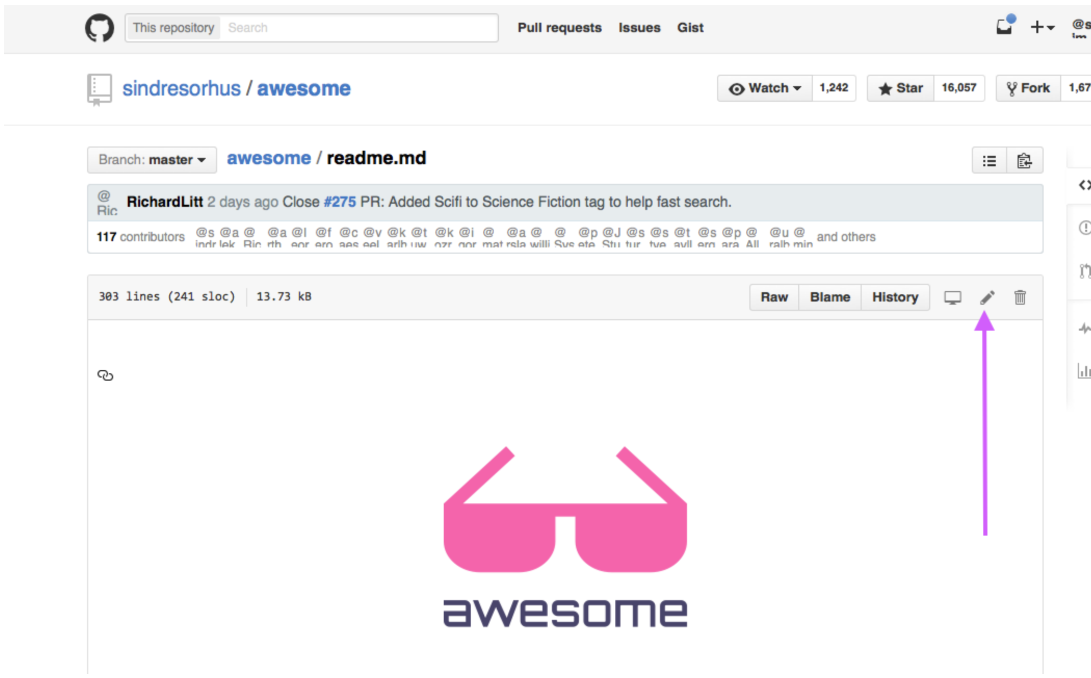
Close #275 PR: Added Scifi to Science Fiction tag to help fast search.

RichardLitt authored 2 days ago
sindresorhus committed 2 days ago
latest commit 41bf87e39a

media	✨ badge ✨	21 days ago
.gitattributes	✨ logo ✨	2 months ago
awesome.md	✨ badge ✨	21 days ago
contributing.md	Update contributing.md	26 days ago
readme.md	Close #275 PR: Added Scifi to Science Fiction tag to help fast search.	2 days ago

readme.md

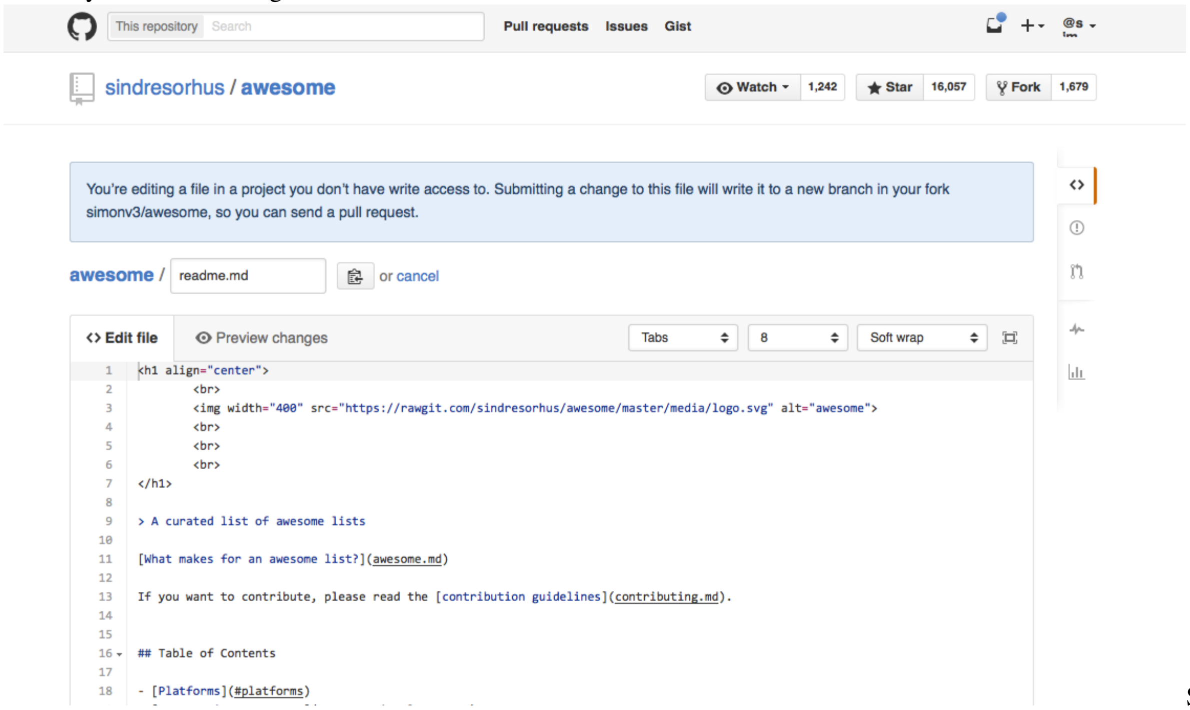
2. Click on the `readme.md` file:
   2 Click on Readme.md



RichardLitt 2 days ago Close #275 PR: Added Scifi to Science Fiction tag to help fast search.

303 lines (241 sloc) | 13.73 kB

3. Now click on the edit icon.
   3 - Click on Edit
4. You can start editing the text of the file in the in-browser editor. Make sure you follow guidelines above. You can use GitHub Flavored Markdown.

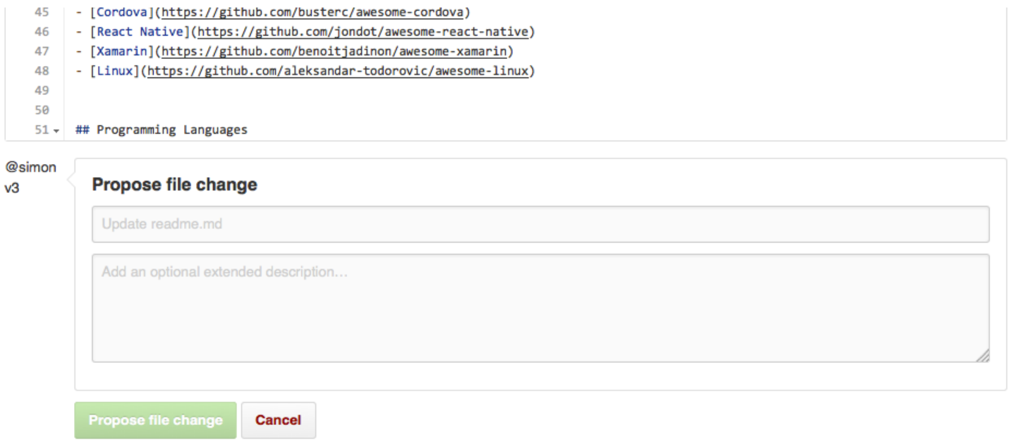


You're editing a file in a project you don't have write access to. Submitting a change to this file will write it to a new branch in your fork simonv3/awesome, so you can send a pull request.

```
<h1 align="center">
	<br>
	<img width="400" src="https://rawgit.com/sindresorhus/awesome/master/media/logo.svg" alt="awesome">
	<br>
	<br>
	<br>
</h1>

> A curated list of awesome lists

[What makes for an awesome list?](awesome.md)

If you want to contribute, please read the [contribution guidelines](contributing.md).


## Table of Contents

- [Platforms](#platforms)
```

Step 4 - Edit the file

5. Say why you're proposing the changes, and then click on "Propose file change".



```
- [Cordova](https://github.com/busterc/awesome-cordova)
- [React Native](https://github.com/jondot/awesome-react-native)
- [Xamarin](https://github.com/benoitjadinon/awesome-xamarin)
- [Linux](https://github.com/aleksandar-todorovic/awesome-linux)


## Programming Languages
```

Step 5 - Propose Changes

6. Submit the pull request!

4.5 Updating your Pull Request

Sometimes, a maintainer of an awesome list will ask you to edit your Pull Request before it is included. This is normally due to spelling errors or because your PR didn't match the awesome-* list guidelines.

Here is a write up on how to change a Pull Request, and the different ways you can do that.

CHAPTER 5

Contributor Covenant Code of Conduct

5.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

5.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

5.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers 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, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

5.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

5.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at florian.kromer@mailbox.org. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

5.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

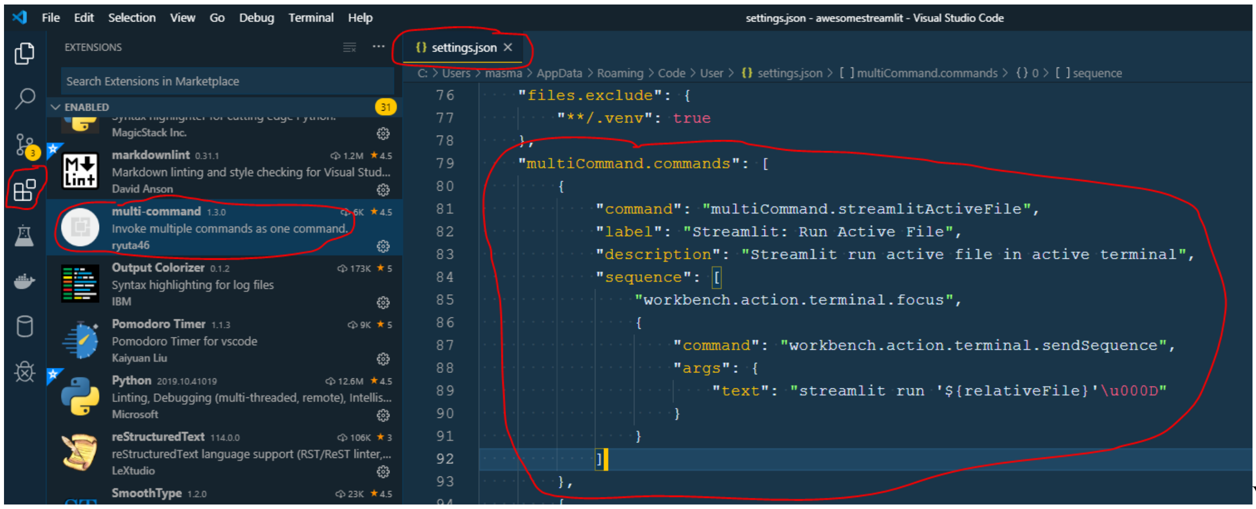
CHAPTER 6

How to use Streamlit with VS Code

6.1 Running Your Streamlit App

You can use the multi-command extension to configure a keyboard short cut to execute `streamlit run <relativeFile.py>`

You start by installing the multi-command extension and adding the configuration shown to your settings.json file.


File
Edit
Selection
View
Go
Debug
Terminal
Help
settings.json - awesomestreamlit - Visual Studio Code
EXTENSIONS
Search Extensions in Marketplace
ENABLED
31
MagicStack Inc.
markdownlint 0.31.1
Markdown linting and style checking for Visual Stud...
David Anson
multi-command 1.3.0
Invoke multiple commands as one command.
ryuta46
Output Colorizer 0.1.2
Syntax highlighting for log files
IBM
Pomodoro Timer 1.1.3
Pomodoro Timer for vscode
Kaiyuan Liu
Python 2019.10.41019
Linting, Debugging (multi-threaded, remote), Intellis...
Microsoft
reStructuredText 114.0.0
reStructuredText language support (RST/ReST linter,...
LeXtudio
SmoothType 1.2.0
settings.json
C: > Users > masma > AppData > Roaming > Code > User > settings.json > [ ] multiCommand.commands > { } 0 > [ ] sequence

```
    "files.exclude": {
        "**/.venv": true
    },
    "multiCommand.commands": [
        {
            "command": "multiCommand.streamlitActiveFile",
            "label": "Streamlit: Run Active File",
            "description": "Streamlit run active file in active terminal",
            "sequence": [
                "workbench.action.terminal.focus",
                {
                    "command": "workbench.action.terminal.sendSequence",
                    "args": {
                        "text": "streamlit run '${relativeFile}'\u000D"
                    }
                }
            ]
        },
```

VS Code multi-command

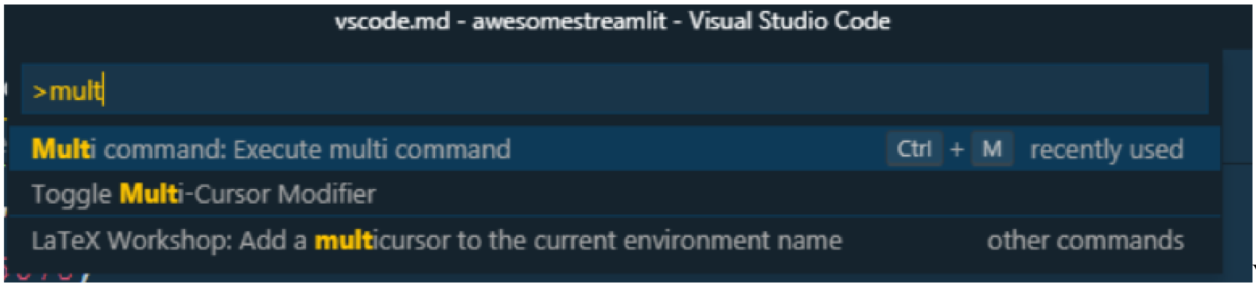
```
{
    "command": "multiCommand.streamlitActiveFile",
    "label": "Streamlit: Run Active File",
    "description": "Streamlit run active file in active terminal",
    "sequence": [
        "workbench.action.terminal.focus",
        {
```

(continues on next page)

(continued from previous page)

```
                "command": "workbench.action.terminal.sendSequence",
                "args": {
                    "text": "streamlit run '${relativeFile}'\u000D"
                }
            }
        ]
},
```

Then you can execute your *streamlit run* command via the command palette (CTRL+SHIFT+P)

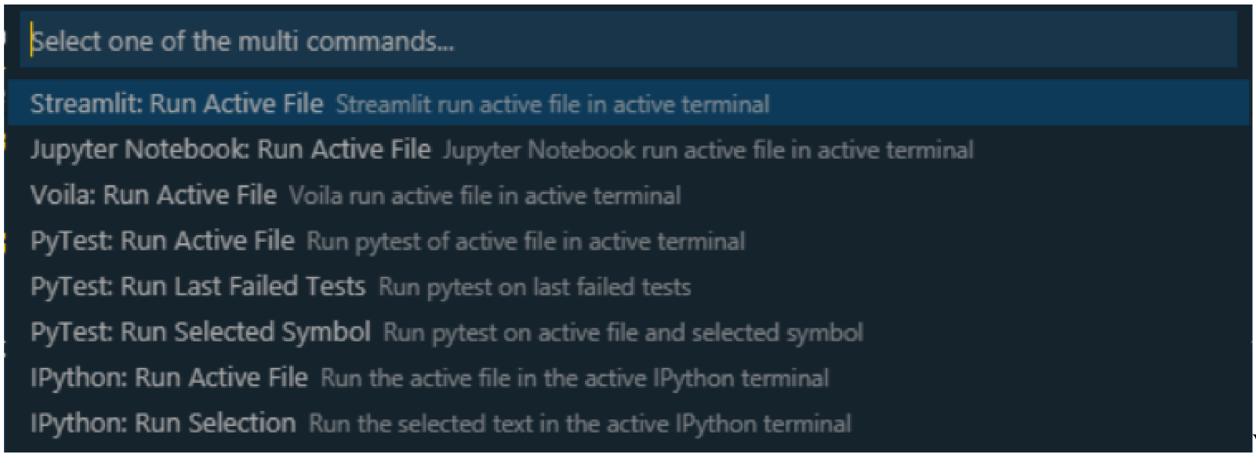

vscode.md - awesomestreamlit - Visual Studio Code

>mult

Multi command: Execute multi command Ctrl + M recently used
Toggle Multi-Cursor Modifier
LaTeX Workshop: Add a multicursor to the current environment name other commands

VS Code multi-command execute

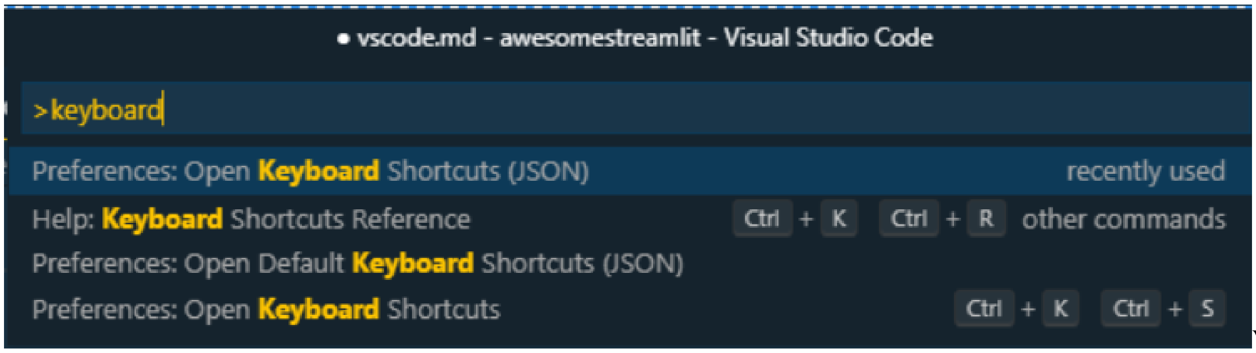
```
{
    "key": "ctrl+m ctrl+s",
    "command": "multiCommand.streamlitActiveFile",
},
```


Select one of the multi commands...

Streamlit: Run Active File Streamlit run active file in active terminal
Jupyter Notebook: Run Active File Jupyter Notebook run active file in active terminal
Voila: Run Active File Voila run active file in active terminal
PyTest: Run Active File Run pytest of active file in active terminal
PyTest: Run Last Failed Tests Run pytest on last failed tests
PyTest: Run Selected Symbol Run pytest on active file and selected symbol
IPython: Run Active File Run the active file in the active IPython terminal
IPython: Run Selection Run the selected text in the active IPython terminal

VS Code multi-command Streamlit Run

Or you can setup a keyboard shortcut in your keybindings.json file to run Streamlit

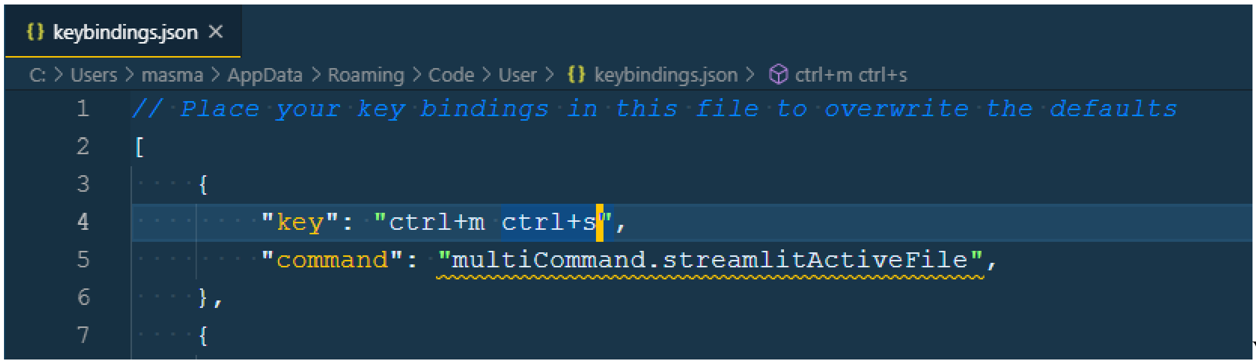

● vscode.md - awesomestreamlit - Visual Studio Code

>keyboard

Preferences: Open Keyboard Shortcuts (JSON) recently used
Help: Keyboard Shortcuts Reference Ctrl + K Ctrl + R other commands
Preferences: Open Default Keyboard Shortcuts (JSON)
Preferences: Open Keyboard Shortcuts Ctrl + K Ctrl + S

VS

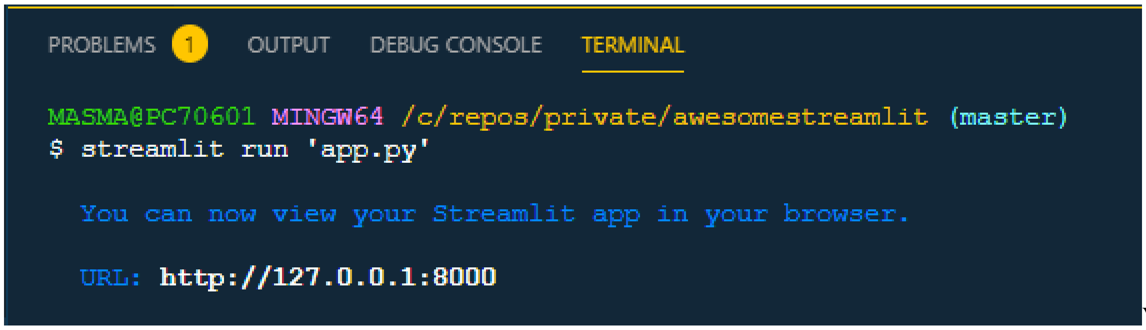
Code open keyboard settings



keybindings.json
C: > Users > masma > AppData > Roaming > Code > User > keybindings.json > ctrl+m ctrl+s

```
// Place your key bindings in this file to overwrite the defaults
[
    {
        "key": "ctrl+m ctrl+s",
        "command": "multiCommand.streamlitActiveFile",
    },
    {
```

VS Code keybindings



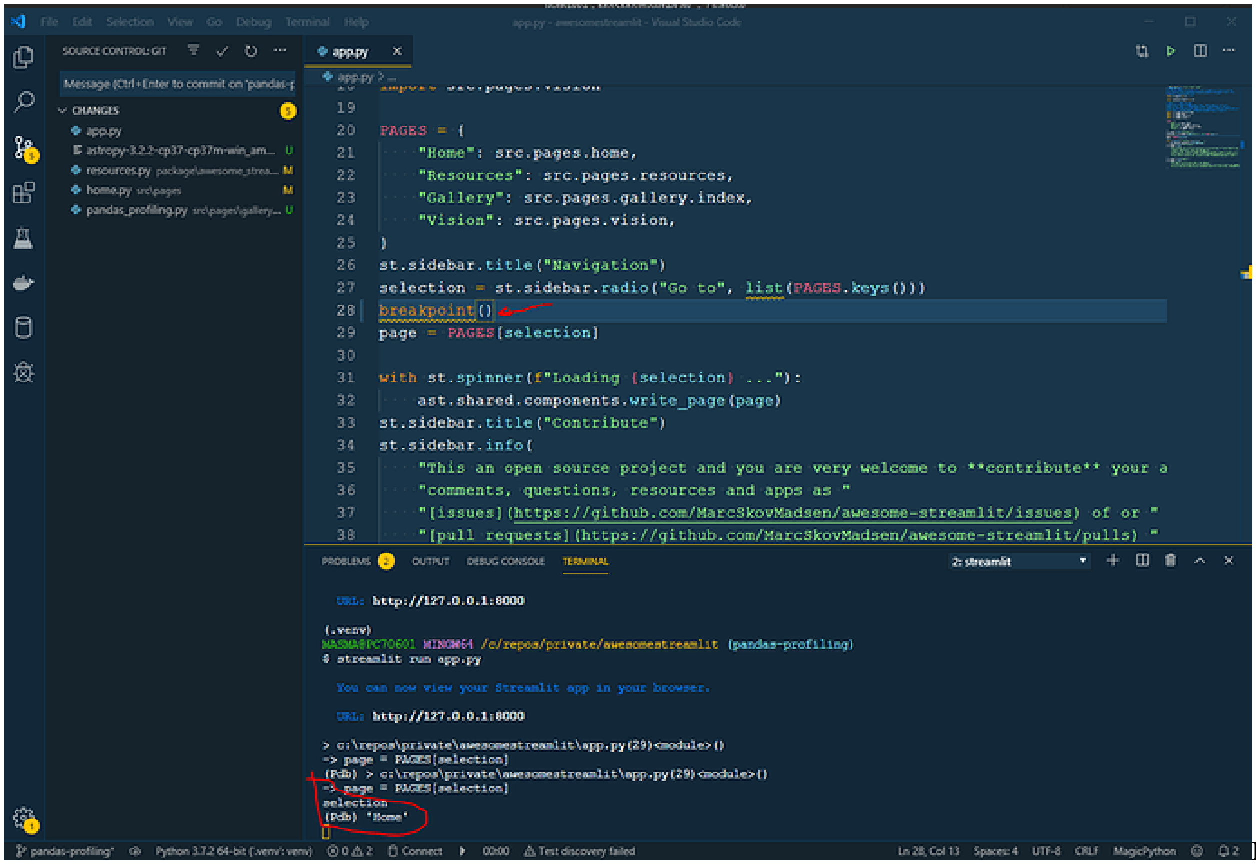
```
MASMA@PC70601 MINGW64 /c/repos/private/awesomestreamlit (master)
$ streamlit run 'app.py'

  You can now view your Streamlit app in your browser.

  URL: http://127.0.0.1:8000
```

VS Code terminal

6.2 Debugging

6.2.1 Manual Debugging

You can **debug mannually** by inserting a `breakpoint()` (Python 3.7+) or `import pdb;pdb.set_trace()` (Python 3.6 or below) in your code.



```
PAGES = {
    "Home": src.pages.home,
    "Resources": src.pages.resources,
    "Gallery": src.pages.gallery.index,
    "Vision": src.pages.vision,
}
st.sidebar.title("Navigation")
selection = st.sidebar.radio("Go to", list(PAGES.keys()))
breakpoint()
page = PAGES[selection]

with st.spinner(f"Loading {selection} ..."):
    ast.shared.components.write_page(page)
st.sidebar.title("Contribute")
st.sidebar.info(
    "This an open source project and you are very welcome to **contribute** your a
    "comments, questions, resources and apps as "
    "[issues](https://github.com/MarcSkovMadsen/awesome-streamlit/issues) of or "
    "[pull requests](https://github.com/MarcSkovMadsen/awesome-streamlit/pulls) "
```

```
  URL: http://127.0.0.1:8000

(.venv)
MASMA@PC70601 MINGW64 /c/repos/private/awesomestreamlit (pandas-profiling)
$ streamlit run app.py

  You can now view your Streamlit app in your browser.

  URL: http://127.0.0.1:8000

> c:\repos\private\awesomestreamlit\app.py(29)<module>()
-> page = PAGES[selection]
(Pdb) > c:\repos\private\awesomestreamlit\app.py(29)<module>()
-> page = PAGES[selection]
selection
(Pdb) 'Home'
```

Debugging via breakpoint

6.2.2 Integrated Debugging

You can also use the **integrated debugger** in VS Code via the ptvsd Python package

Please note that andaag reported the below to not work on ubuntu 18.04.3 LTS with Python 3.6.8. He gets a `ValueError: signal only works in main thread` error. See issue 648. It's working really well for me on Windows with Python 3.7.4 though.

First you should `pip install ptvsd`.

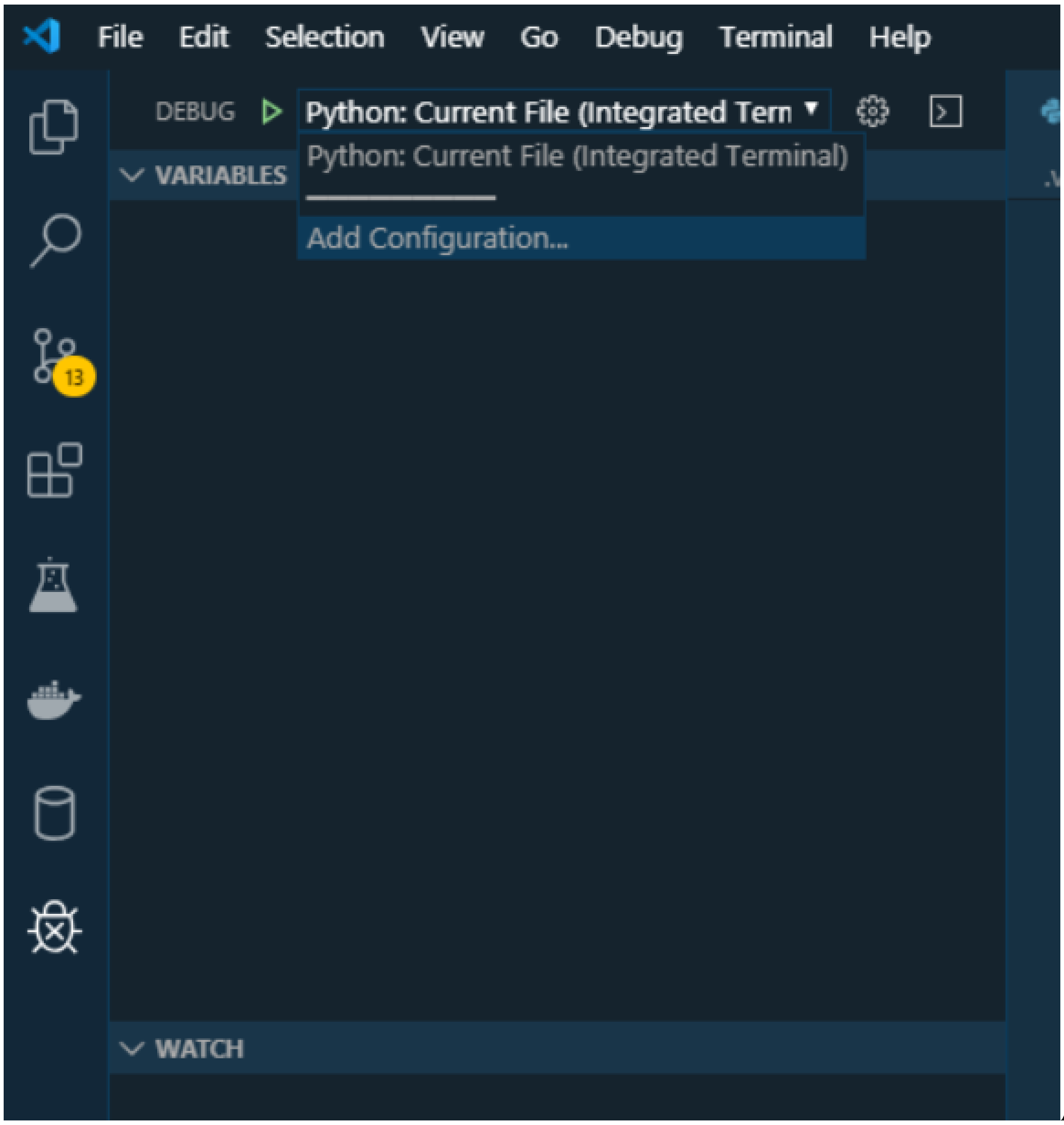
Then you need to insert the following snippet in your `<your-app_name>.py` file.

```
import ptvsd
ptvsd.enable_attach(address=('localhost', 5678))
ptvsd.wait_for_attach() # Only include this line if you always wan't to attach the␣
↪debugger
```

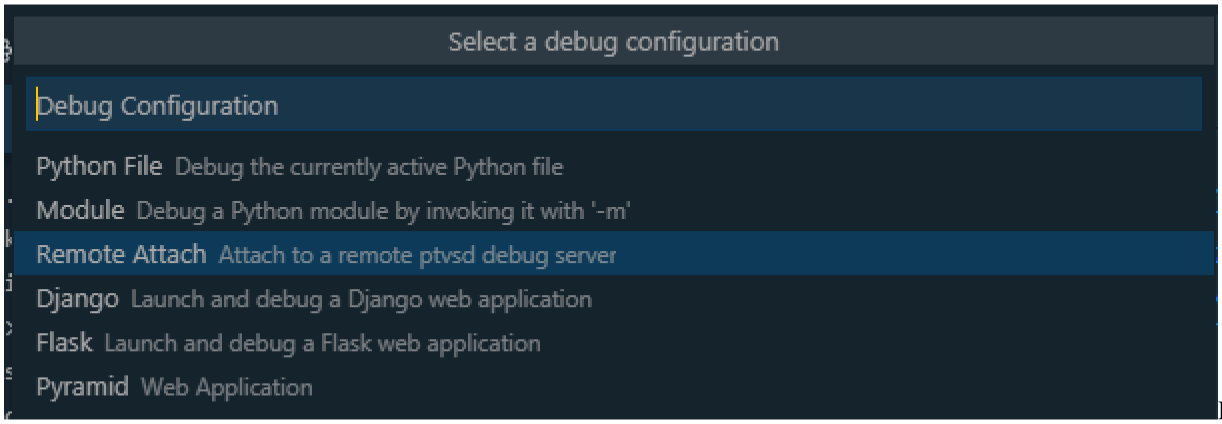
Then you can start your Streamlit app

```
streamlit run <your-app_name>.py
```

Then you should configure your *Remote Attach: debug PTVSD option*



Add Configuration

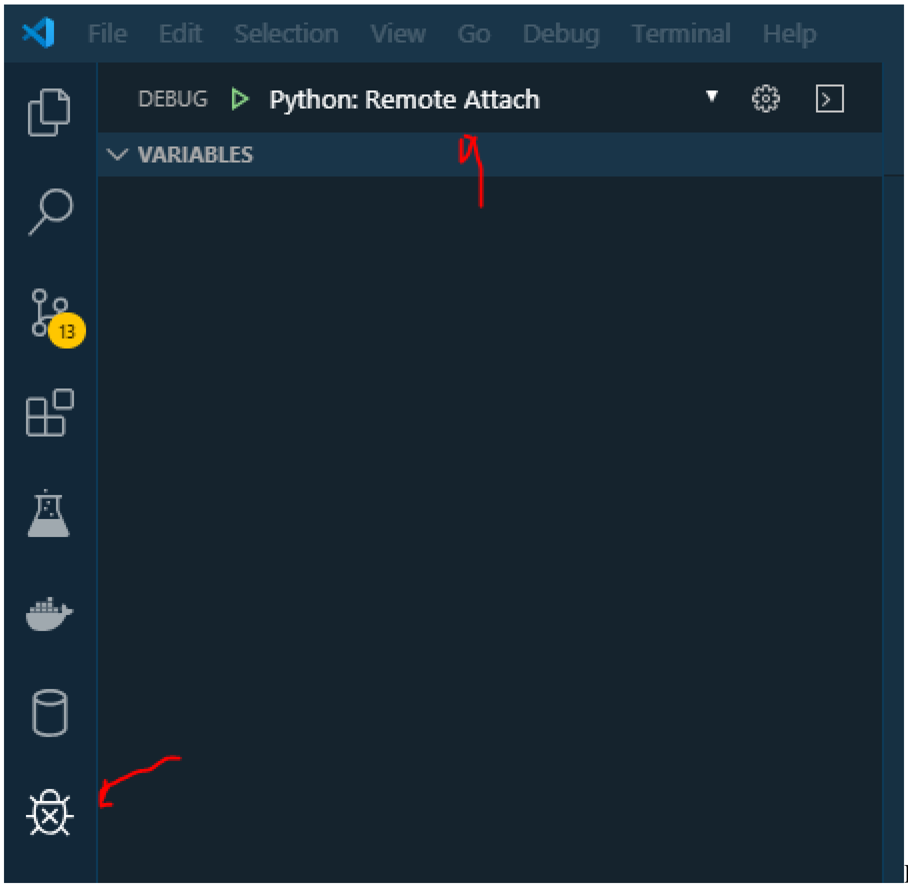

Select a debug configuration
Debug Configuration
Python File Debug the currently active Python file
Module Debug a Python module by invoking it with '-m'
Remote Attach Attach to a remote ptvsd debug server
Django Launch and debug a Django web application
Flask Launch and debug a Flask web application
Pyramid Web Application
Debug Configuration

and update to the below in your launch.json file. Please make sure that you manually insert the *redirectOutput* setting below.

```
{
    "name": "Python: Remote Attach",
    "type": "python",
    "request": "attach",
    "port": 5678,
    "host": "localhost",
    "justMyCode": true,
    "redirectOutput": true,
    "pathMappings": [
        {
            "localRoot": "${workspaceFolder}",
            "remoteRoot": "."
        }
    ]
}
```

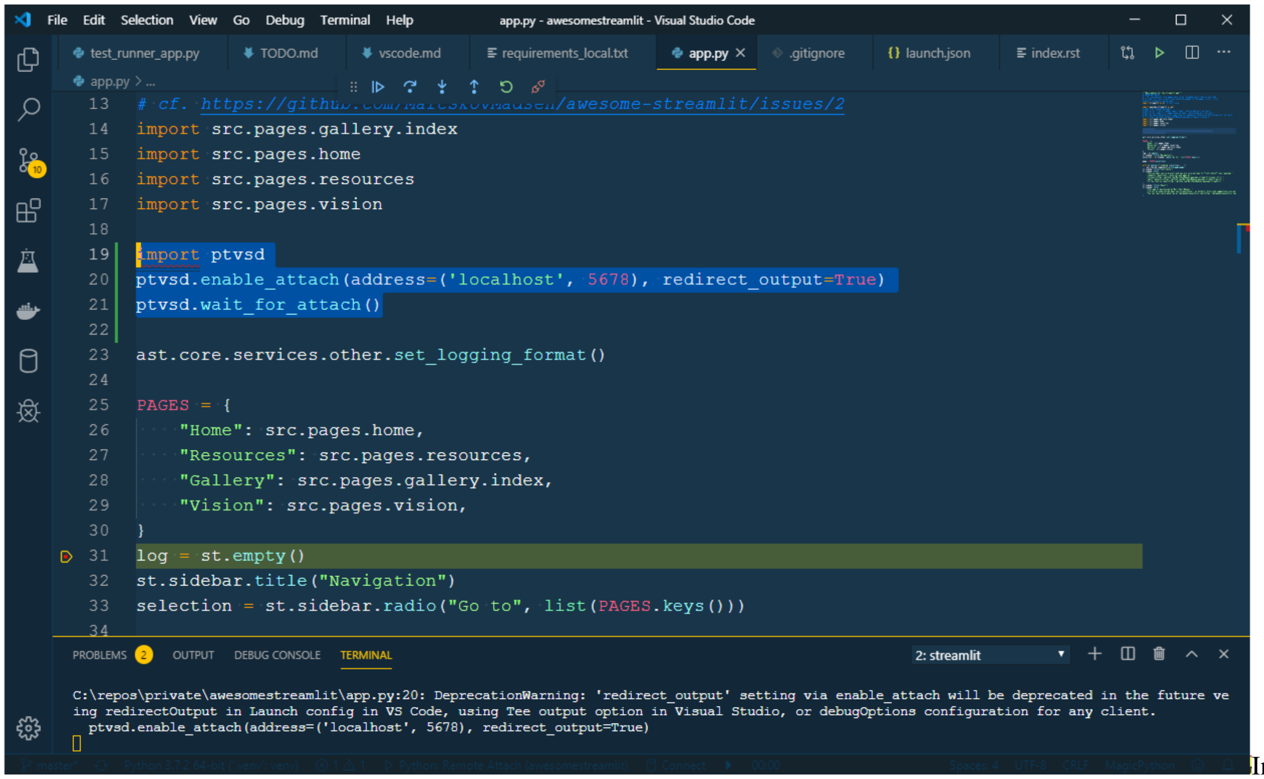
Please note that by default you will be debugging your own code only. If you wan't to debug into for example the streamlit code, then you can change the `justMyCode` setting from `true` to `false`.

Finally you can attach the debugger by clicking the debugger play button



Python remote attach

and you can debug away.



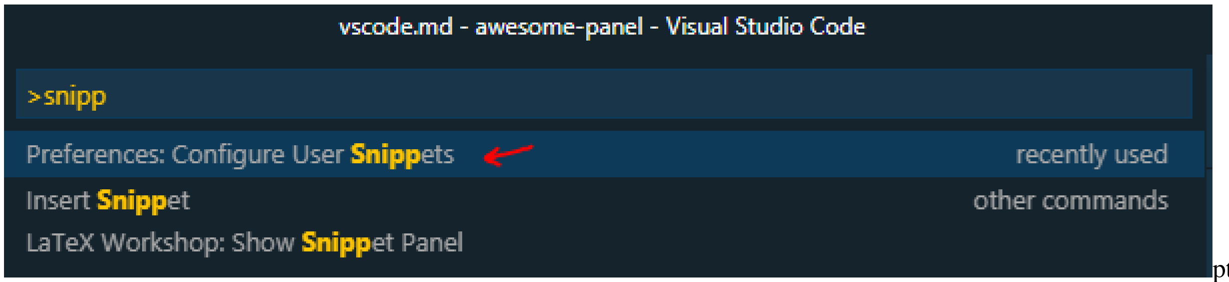
```python
# cf. https://github.com/MarcSkovMadsen/awesome-streamlit/issues/2
import src.pages.gallery.index
import src.pages.home
import src.pages.resources
import src.pages.vision

import ptvsd
ptvsd.enable_attach(address=('localhost', 5678), redirect_output=True)
ptvsd.wait_for_attach()

ast.core.services.other.set_logging_format()

PAGES = {
    "Home": src.pages.home,
    "Resources": src.pages.resources,
    "Gallery": src.pages.gallery.index,
    "Vision": src.pages.vision,
}
log = st.empty()
st.sidebar.title("Navigation")
selection = st.sidebar.radio("Go to", list(PAGES.keys()))
```

```
C:\repos\private\awesomestreamlit\app.py:20: DeprecationWarning: 'redirect_output' setting via enable_attach will be deprecated in the future ve
ing redirectOutput in Launch config in VS Code, using Tee output option in Visual Studio, or debugOptions configuration for any client.
  ptvsd.enable_attach(address=('localhost', 5678), redirect_output=True)
```

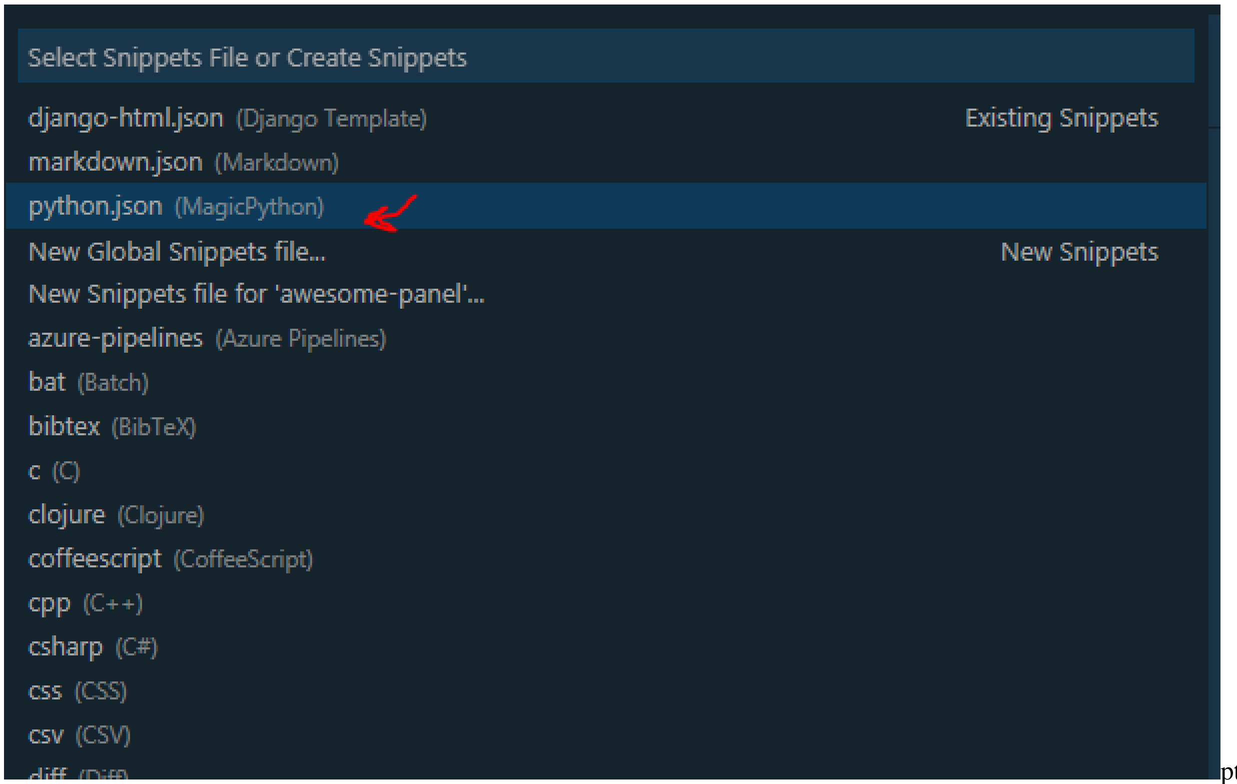
Integrated Debugger

Using a ptvsd snippet

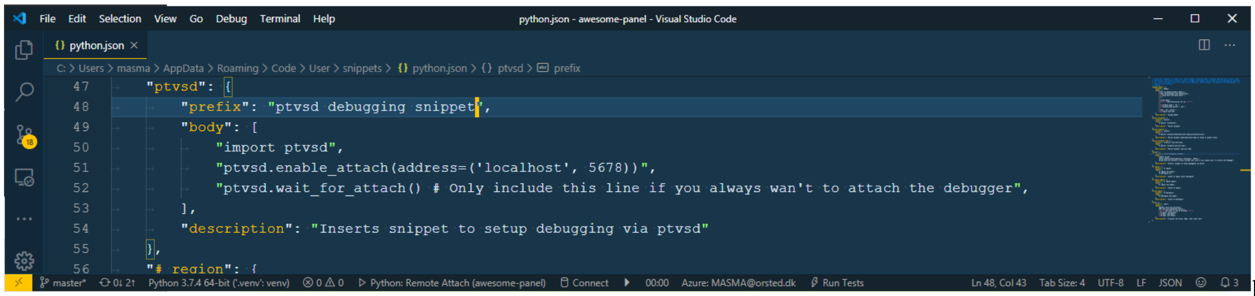
You can create a snippet in your `python.json` snippet configuration to insert the `import ptvsd...` code.


vscode.md - awesome-panel - Visual Studio Code
>snipp
Preferences: Configure User Snippets
recently used
Insert Snippet
other commands
LaTeX Workshop: Show Snippet Panel

ptvsd code snippet


Select Snippets File or Create Snippets
django-html.json (Django Template)
Existing Snippets
markdown.json (Markdown)
python.json (MagicPython)
New Global Snippets file...
New Snippets
New Snippets file for 'awesome-panel'...
azure-pipelines (Azure Pipelines)
bat (Batch)
bibtex (BibTeX)
c (C)
clojure (Clojure)
coffeescript (CoffeeScript)
cpp (C++)
csharp (C#)
css (CSS)
csv (CSV)
diff (Diff)

ptvsd code snippet


File Edit Selection View Go Debug Terminal Help
python.json - awesome-panel - Visual Studio Code
python.json
C: > Users > masma > AppData > Roaming > Code > User > snippets > python.json > ptvsd > prefix
master* Python 3.7.4 64-bit ('.venv': venv) Python: Remote Attach (awesome-panel) Connect Azure: MASMA@orsted.dk Run Tests Ln 48, Col 43 Tab Size: 4 UTF-8 LF JSON

```
    "ptvsd": {
        "prefix": "ptvsd debugging snippet",
        "body": [
            "import ptvsd",
            "ptvsd.enable_attach(address=('localhost', 5678))",
            "ptvsd.wait_for_attach() # Only include this line if you always wan't to attach the debugger",
        ],
        "description": "Inserts snippet to setup debugging via ptvsd"
    },
    "# region": {
```

ptvsd code snippet

```
"ptvsd": {
    "prefix": "ptvsd debugging snippet",
    "body": [
        "import ptvsd",
        "ptvsd.enable_attach(address=('localhost', 5678))",
        "print('Ready to attach the VS Code debugger')",
        "ptvsd.wait_for_attach() # Only include this line if you always wan't to␣
↪attach the debugger",
    ],
    "description": "Inserts snippet to setup debugging via ptvsd"
},
```

Using a dedicated app_debug_vscode.py file for debugging

Adding and removing the *ptvsd* code above can be cumbersome. So a usefull trick is to setup a dedicated *app_debug_vscode.py* file for debugging.

Assuming your app.py file has a `def main():` function, then your *app_debug_vscode.py* file could look as follows

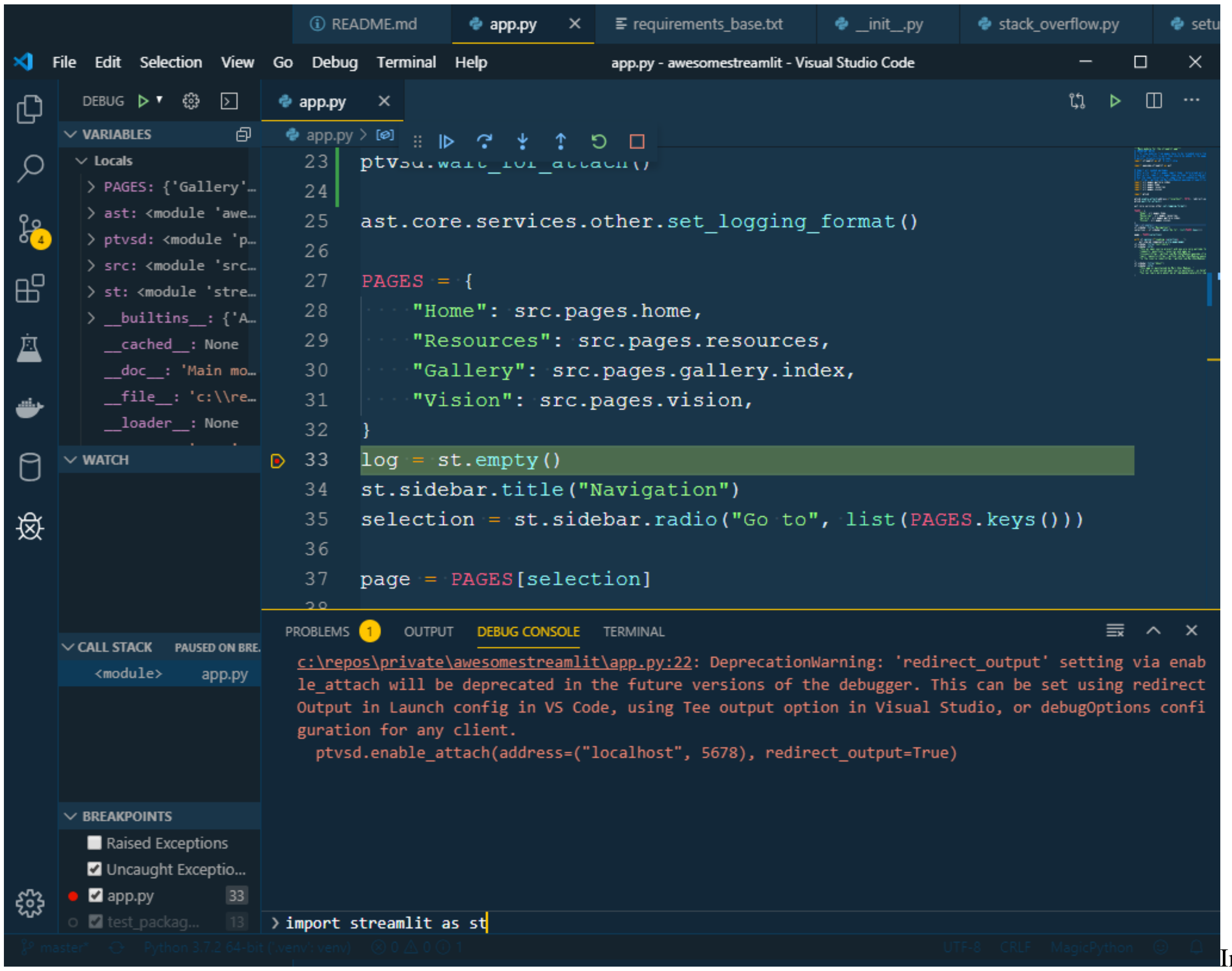
```python
"""Use this module for development with VS Code and the integrated debugger"""
import ptvsd
import streamlit as st

import app

# pylint: disable=invalid-name
markdown = st.markdown(
    """
## Ready to attach the VS Code Debugger!

![Python: Remote Attach](https://awesome-streamlit.readthedocs.io/en/latest/_images/
↪vscode_python_remote_attach.png)

for more info see the [VS Code section at awesome-streamlit.readthedocs.io]
(https://awesome-streamlit.readthedocs.io/en/latest/vscode.html#integrated-debugging)
"""
)


ptvsd.enable_attach(address=("localhost", 5678))
ptvsd.wait_for_attach()

markdown.empty()

app.main()
```

then you run `streamlit run app_debug_vscode.py` instead of `streamlit run app.py` and attach the debugger.

For a use case see my app.py and app_dev_vscode.py files.

Using the integrated Debugging Console

When you are running your integrated debugging in VS Code, you can use the *Debugging Console* with Streamlit if you `import streamlit as st`. Then you can write dataframes and charts to the browser window and take a better look at your data, than you can in VS Code.



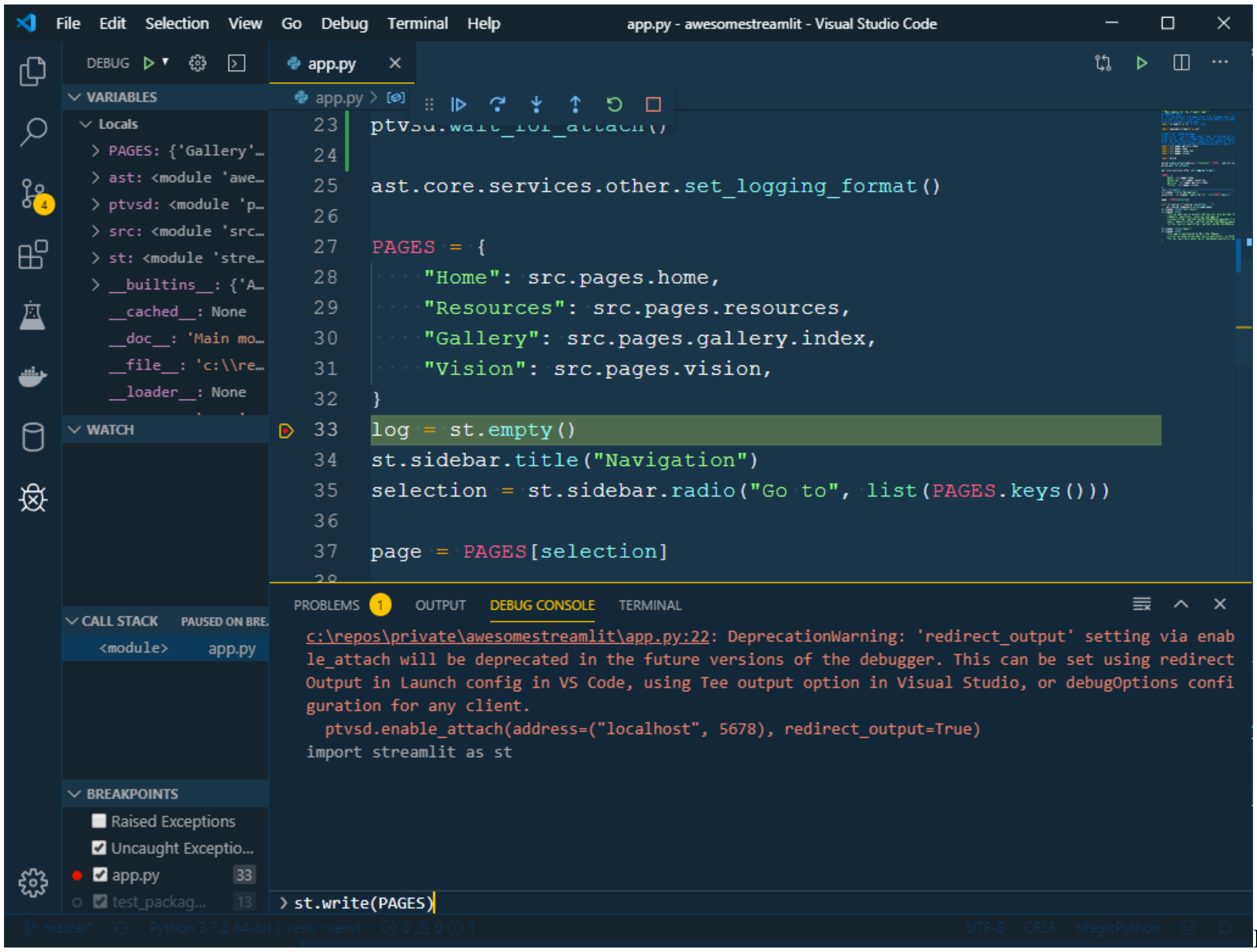
```
ptvsd.wait_for_attach()

ast.core.services.other.set_logging_format()

PAGES = {
    "Home": src.pages.home,
    "Resources": src.pages.resources,
    "Gallery": src.pages.gallery.index,
    "Vision": src.pages.vision,
}
log = st.empty()
st.sidebar.title("Navigation")
selection = st.sidebar.radio("Go to", list(PAGES.keys()))

page = PAGES[selection]
```

```
c:\repos\private\awesomestreamlit\app.py:22: DeprecationWarning: 'redirect_output' setting via enab
le_attach will be deprecated in the future versions of the debugger. This can be set using redirect
Output in Launch config in VS Code, using Tee output option in Visual Studio, or debugOptions confi
guration for any client.
  ptvsd.enable_attach(address=("localhost", 5678), redirect_output=True)
```

```
> import streamlit as st
```

Import

Streamlit



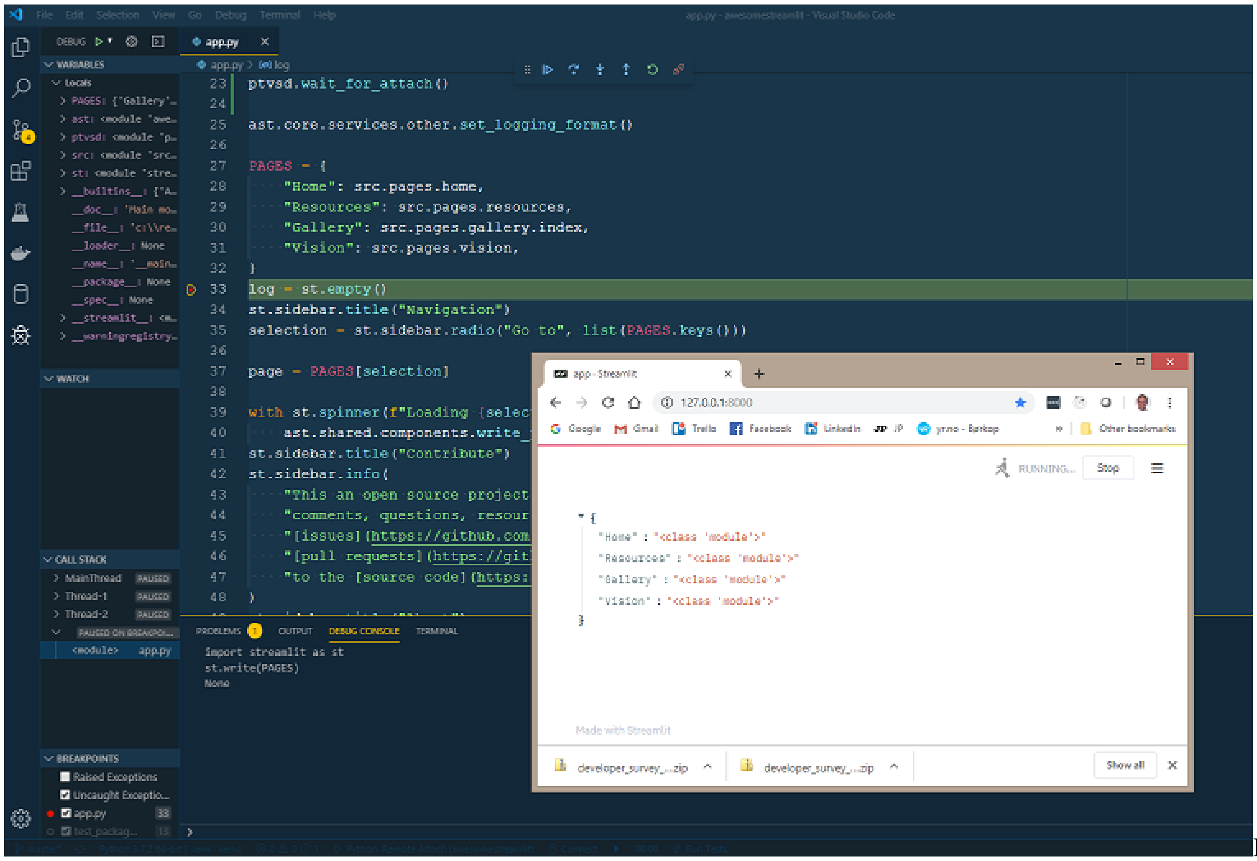
```
ptvsd.wait_for_attach()

ast.core.services.other.set_logging_format()

PAGES = {
    "Home": src.pages.home,
    "Resources": src.pages.resources,
    "Gallery": src.pages.gallery.index,
    "Vision": src.pages.vision,
}
log = st.empty()
st.sidebar.title("Navigation")
selection = st.sidebar.radio("Go to", list(PAGES.keys()))

page = PAGES[selection]
```

```
Locals
> PAGES: {'Gallery'…
> ast: <module 'awe…
> ptvsd: <module 'p…
> src: <module 'src…
> st: <module 'stre…
> __builtins__: {'A…
  __cached__: None
  __doc__: 'Main mo…
  __file__: 'c:\\re…
  __loader__: None
```

```
c:\repos\private\awesomestreamlit\app.py:22: DeprecationWarning: 'redirect_output' setting via enable_attach will be deprecated in the future versions of the debugger. This can be set using redirect Output in Launch config in VS Code, using Tee output option in Visual Studio, or debugOptions configuration for any client.
  ptvsd.enable_attach(address=("localhost", 5678), redirect_output=True)
import streamlit as st
```

```
> st.write(PAGES)
```

Import

Streamlit 

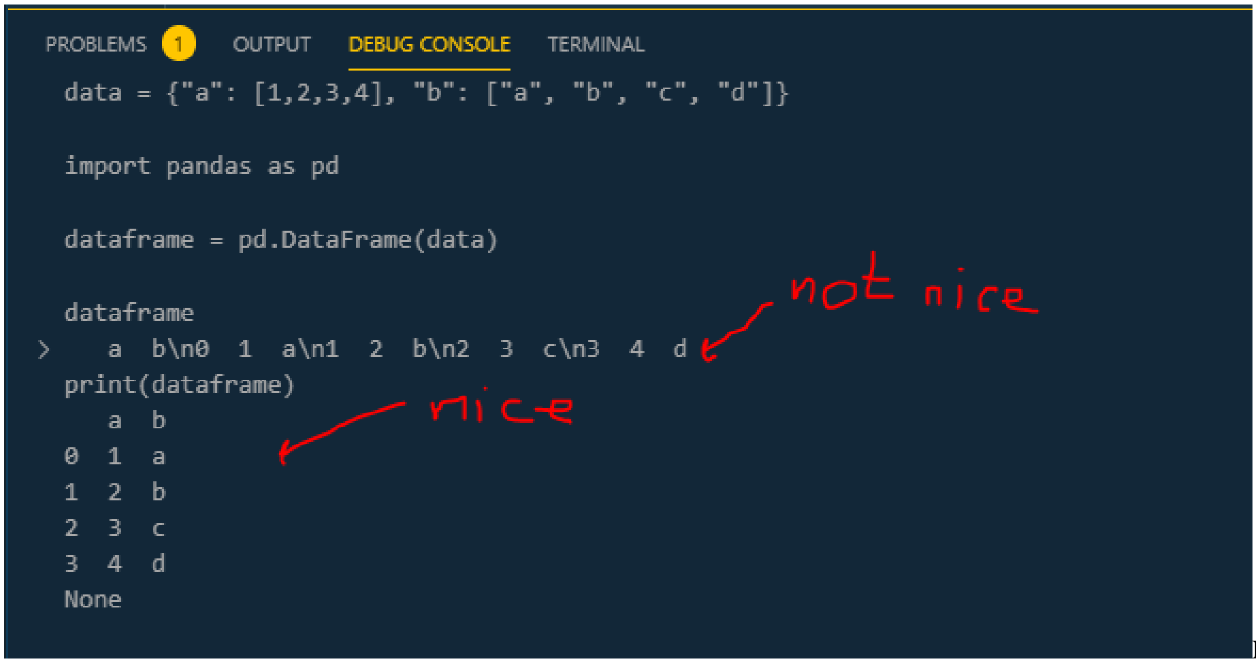
```
ptvsd.wait_for_attach()

ast.core.services.other.set_logging_format()

PAGES = {
    "Home": src.pages.home,
    "Resources": src.pages.resources,
    "Gallery": src.pages.gallery.index,
    "Vision": src.pages.vision,
}
log = st.empty()
st.sidebar.title("Navigation")
selection = st.sidebar.radio("Go to", list(PAGES.keys()))

page = PAGES[selection]

with st.spinner(f"Loading {selec
    ast.shared.components.write_
st.sidebar.title("Contribute")
st.sidebar.info(
    "This an open source project
    "comments, questions, resour
    "[issues](https://github.com
    "[pull requests](https://git
    "to the [source code](https:
)
```

```
import streamlit as st
st.write(PAGES)
None
```

```
{
  "Home" : "<class 'module'>"
  "Resources" : "<class 'module'>"
  "Gallery" : "<class 'module'>"
  "Vision" : "<class 'module'>"
}
```

Import Streamlit

You should also remember to *print* your dataframes to the debugger console to get a nice formatting.



```
data = {"a": [1,2,3,4], "b": ["a", "b", "c", "d"]}

import pandas as pd

dataframe = pd.DataFrame(data)

dataframe
>    a  b\n0  1  a\n1  2  b\n2  3  c\n3  4  d
print(dataframe)
   a  b
0  1  a
1  2  b
2  3  c
3  4  d
None
```

Nice Print of DataFrame

6.3 Installing the Streamlit Repo for Development using Visual Studio Online

I would like to experiment with and maybe contribute to the streamlit/streamlit repo.

There is an official Streamlit Contributing guide I can follow to get up and running.

But I'm running on Windows 8.1 so it's a it problematic getting it up and running.

- I cannot install directly on Windows because the Makefile and associated tools are very dependent on Linux.
- I cannot use *Visual Studio Code Remote - Containers* as described here because it requires Windows 10 (not 8.1) and Docker (not Docker Toolbox)
- Maybe I could have installed Linux on my machine. But . . .
- I guess I could install in a Docker container locally but not integrate with VS Code. But that would not be efficient.

I then settled on trying out the new Visual Studio Online experience. I got it working successfully after some time. Below I will describe the steps.

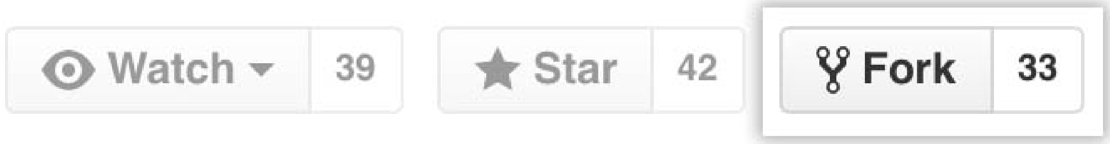
DISCLAIMER: THE BELOW WAS SO COMPLICATED AND HAD TO BE REPEATED SEVERAL TIMES THAT I CANNOT GUARENTEE THAT IT WILL WORK 100%. BUT I HOPE IT STILL HELPS A LOT.

IF SOMETHING GOES WRONG TAKE A LOOK AT THE TROUBLE SHOOTING SECTION BELOW.

6.3.1 Forking the Streamlit Repo

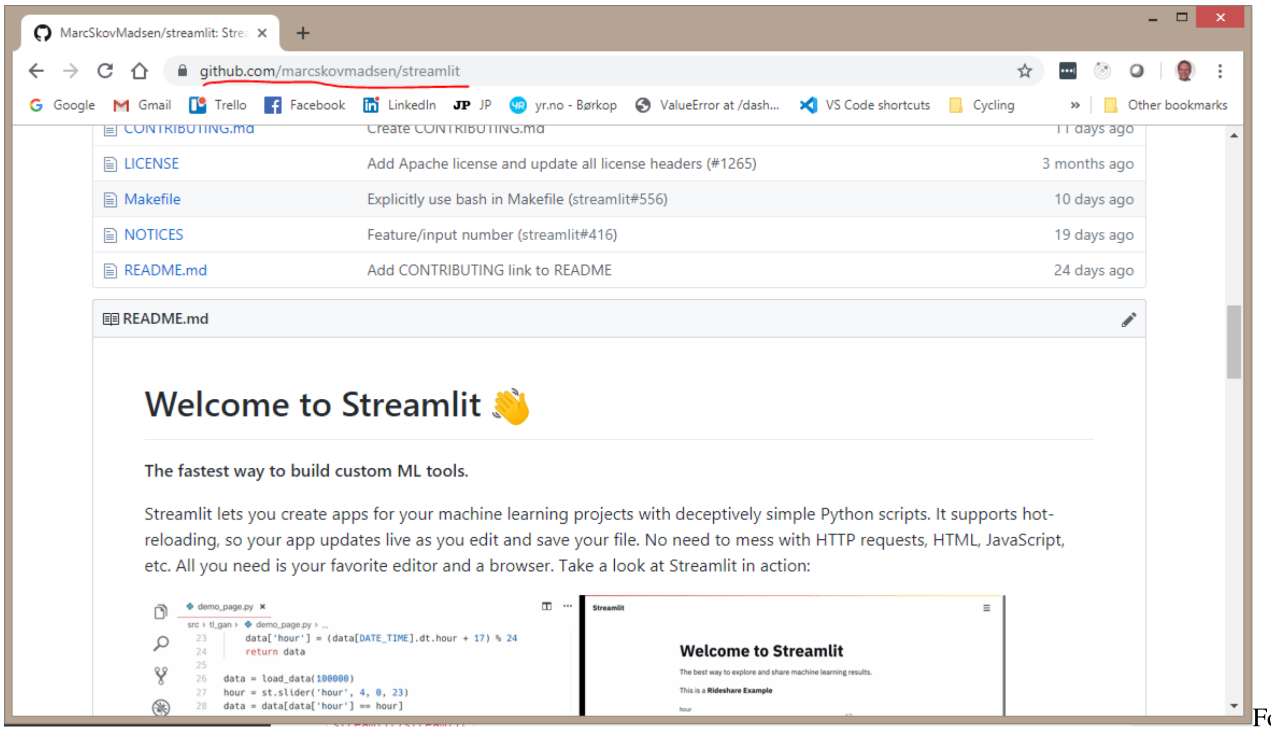
Forking a repository is a simple two-step process.

1 On GitHub navigate to the streamlit/streamlit repository 2 In the top-right corner of the page, click Fork



Fork Button

That's it! Now I have a fork of the original streamlit/streamlit repository.


MarcSkovMadsen/streamlit: Stre

github.com/marcskovmadsen/streamlit

Google Gmail Trello Facebook LinkedIn JP yr.no - Børkop ValueError at /dash... VS Code shortcuts Cycling Other bookmarks

CONTRIBUTING.md	Create CONTRIBUTING.md	11 days ago
LICENSE	Add Apache license and update all license headers (#1265)	3 months ago
Makefile	Explicitly use bash in Makefile (streamlit#556)	10 days ago
NOTICES	Feature/input number (streamlit#416)	19 days ago
README.md	Add CONTRIBUTING link to README	24 days ago

README.md

Welcome to Streamlit 👋

The fastest way to build custom ML tools.

Streamlit lets you create apps for your machine learning projects with deceptively simple Python scripts. It supports hot-reloading, so your app updates live as you edit and save your file. No need to mess with HTTP requests, HTML, JavaScript, etc. All you need is your favorite editor and a browser. Take a look at Streamlit in action:

```
    data['hour'] = (data[DATE_TIME].dt.hour + 17) % 24
    return data

data = load_data(100000)
hour = st.slider('hour', 4, 0, 23)
data = data[data['hour'] == hour]
```

Welcome to Streamlit

The best way to explore and share machine learning results.

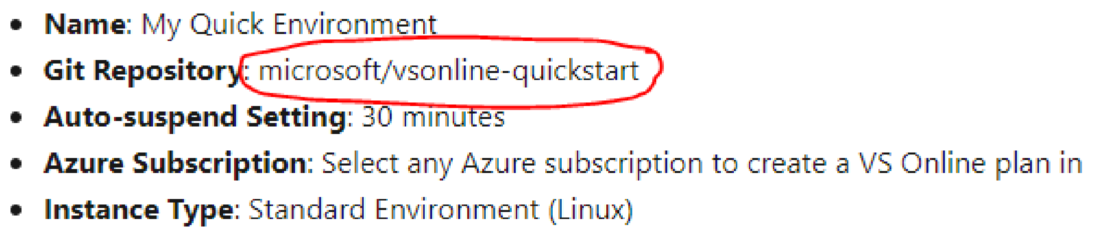
This is a Rideshare Example

Fork of Streamlit

6.3.2 Setting up the Streamlit Repo as a a Visual Studio Online Environment

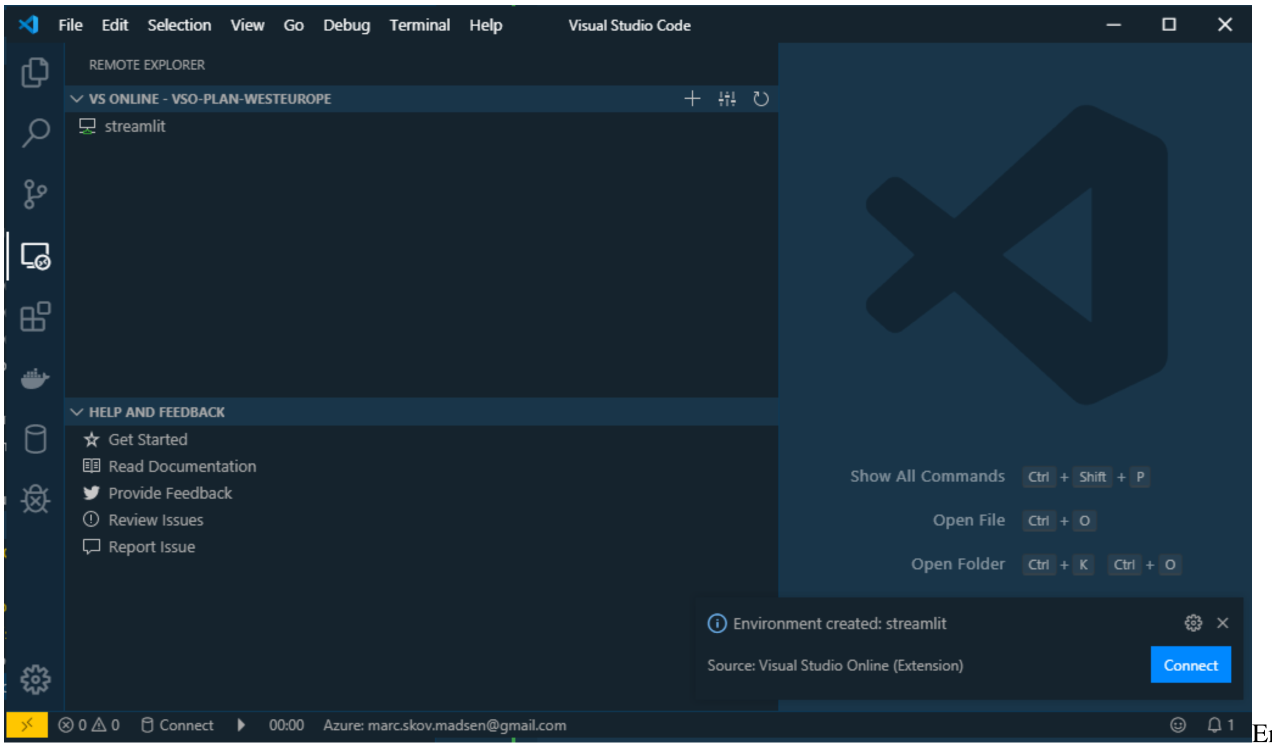
Visual Studio Online is a new experiment for working in VS Code locally, but running everything inside a docker container environment in the cloud.

To create a docker container environment containing the streamlit repo I followed step 1-4 of the offical VS Code Quickstart Guide. But I replaced the `microsoft/vsonline-quickstart` with `<my-user-name>/streamlit`.


- **Name**: My Quick Environment
- **Git Repository**: microsoft/vsonline-quickstart
- **Auto-suspend Setting**: 30 minutes
- **Azure Subscription**: Select any Azure subscription to create a VS Online plan in
- **Instance Type**: Standard Environment (Linux)

Replicate with streamlit/streamlit

Finally the environment was created

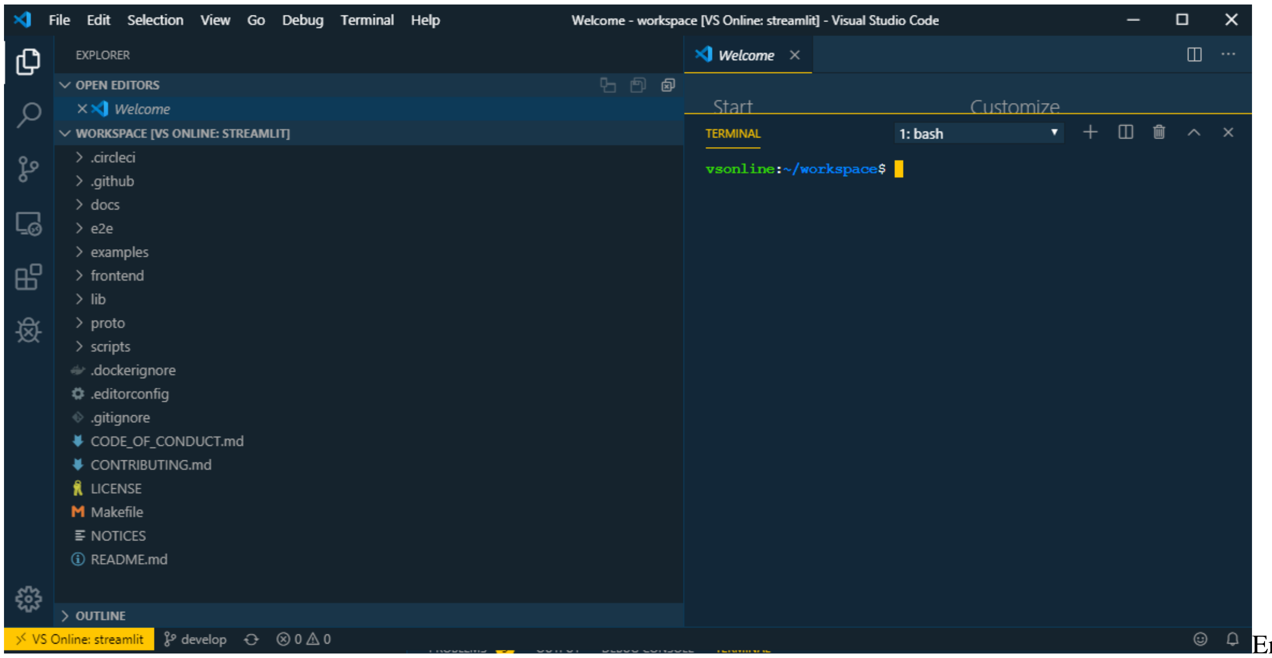

File Edit Selection View Go Debug Terminal Help
Visual Studio Code
REMOTE EXPLORER
VS ONLINE - VSO-PLAN-WESTEUROPE
streamlit
HELP AND FEEDBACK
Get Started
Read Documentation
Provide Feedback
Review Issues
Report Issue
Show All Commands Ctrl + Shift + P
Open File Ctrl + O
Open Folder Ctrl + K Ctrl + O
Connect
Azure: marc.skov.madsen@gmail.com

Environment created: streamlit

Source: Visual Studio Online (Extension)

Environment created

and I click the connect button and it connects.



```
vsonline:~/workspace$
```

Environment connected

6.3.3 Were running Python 3.5.3 on Debian!

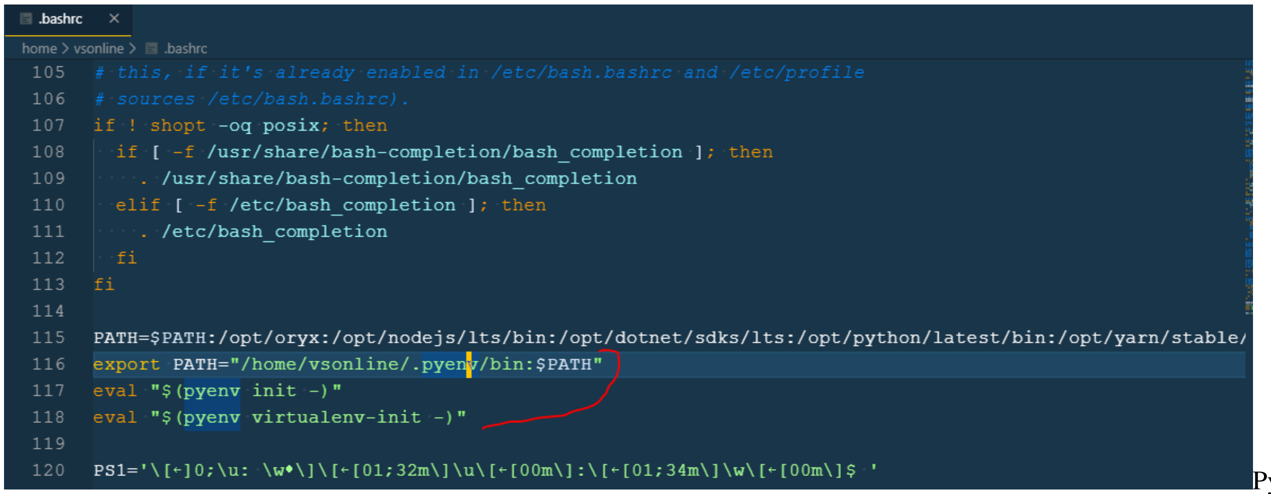
It's nice to see what is in the environment

```
vsonline:~/workspace$ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
vsonline:~/workspace$ python3
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
vsonline:~/workspace$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
NAME="Debian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
VERSION_CODENAME=stretch
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
vsonline:~/workspace/Python-3.7.4$ nproc
4
```

6.3.4 Set Up My Base Environment

I then followed the Ubuntu section with

- `sudo apt install graphviz python3-distutils` replaced by `sudo apt install graphviz`
  - See the troubleshouting guide wrt `python3-distutils`.
- and pyenv added to the path of the ~/.bashrc file



```
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

PATH=$PATH:/opt/oryx:/opt/nodejs/lts/bin:/opt/dotnet/sdks/lts:/opt/python/latest/bin:/opt/yarn/stable/
export PATH="/home/vsonline/.pyenv/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"

PS1='\[←]0;\u: \w•\]\[←[01;32m\]\u\[←[00m\]:\[←[01;34m\]\w\[←[00m\]$ '
```

Pyenv and .bashrc

6.3.5 Installing Python 3.7.4

I used pyenv to install Python 3.7.4

```
vsonline:~/workspace/lib$ pyenv install 3.7.4
```

6.3.6 Grab the code

Nothing to do here as it's already installed.

6.3.7 Create a new Python environment

```
vsonline:~/workspace/lib$ pyenv global 3.7.4
vsonline:~/workspace/lib$ python3
Python 3.7.4 (default, Nov 10 2019, 11:50:45)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
vsonline:~/workspace/lib$ python3 -m venv ~/.venv
vsonline:~/workspace/lib$ source ~/.venv/bin/activate
```

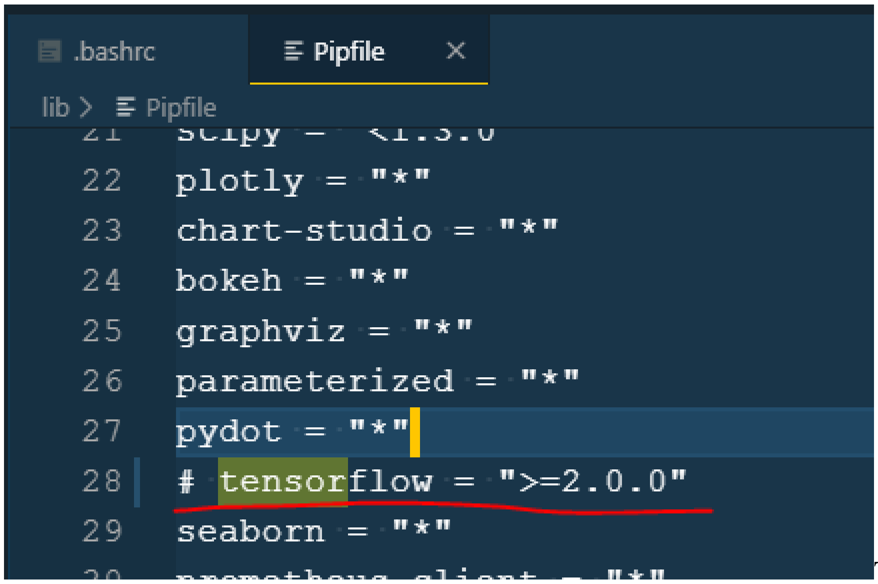
```
(.venv) vsonline:~/workspace/lib$ pipenv install
Creating a virtualenv for this project...
Pipfile: /home/vsonline/workspace/lib/Pipfile
Using /home/vsonline/.pyenv/versions/3.7.4/bin/python3.7 (3.7.4) to create␣
↪virtualenv...
 Creating virtual environment...Already using interpreter /home/vsonline/.pyenv/
↪versions/3.7.4/bin/python3.7
Using base prefix '/home/vsonline/.pyenv/versions/3.7.4'
New python executable in /home/vsonline/.local/share/virtualenvs/lib-vom9Vlgm/bin/
↪python3.7
Also creating executable in /home/vsonline/.local/share/virtualenvs/lib-vom9Vlgm/bin/
↪python
Installing setuptools, pip, wheel...
done.

✓ Successfully created virtual environment!
Virtualenv location: /home/vsonline/.local/share/virtualenvs/lib-vom9Vlgm
```

activate the environment

```
pipenv shell
```

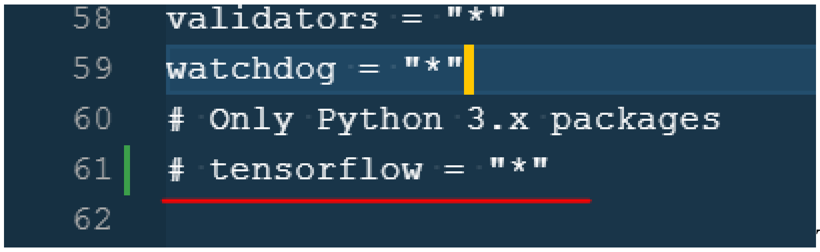
6.3.8 Onetime setup

Disable Tensorflow in the Pipfile



```
.bashrc    Pipfile
lib > Pipfile
plotly = "*"
chart-studio = "*"
bokeh = "*"
graphviz = "*"
parameterized = "*"
pydot = "*"
# tensorflow = ">=2.0.0"
seaborn = "*"
```

Tensorflow removed from pipenv file



```
validators = "*"
watchdog = "*"
# Only Python 3.x packages
# tensorflow = "*"
```

Tensorflow removed from pipenv file

and install it manually

```
pip install tensorflow>=2.0.0
```

then run

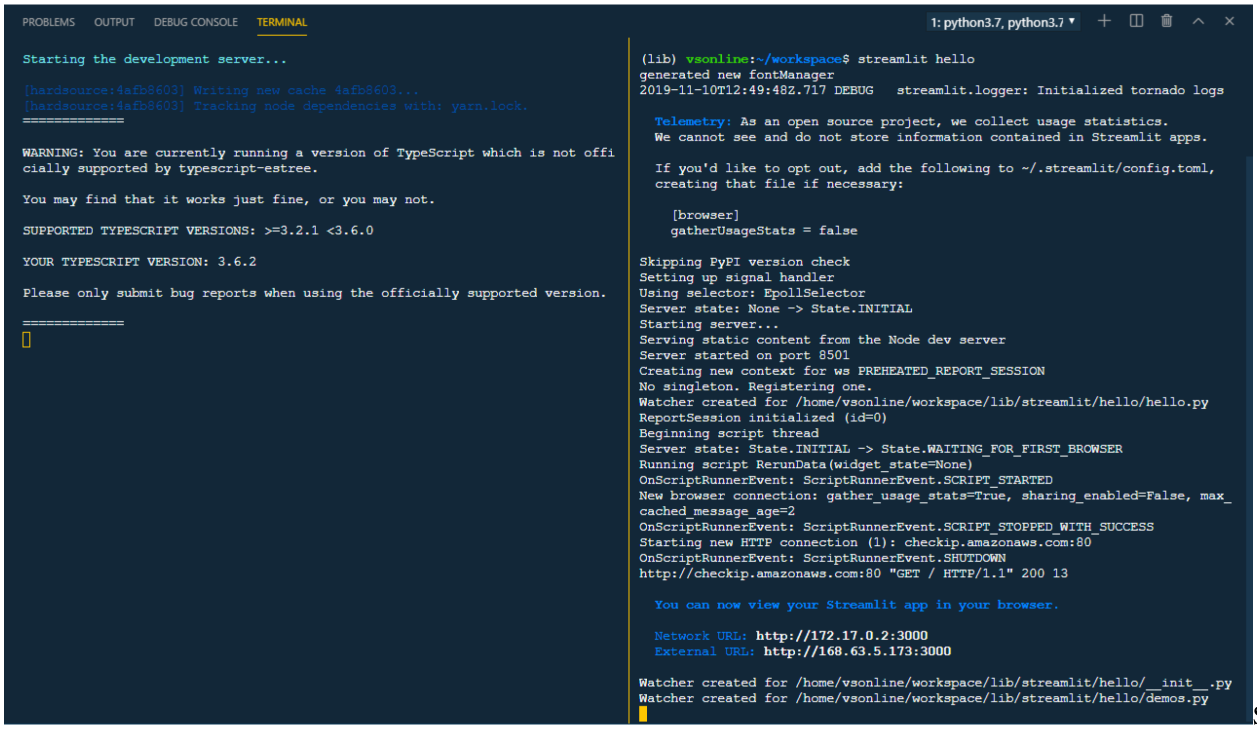
```
make all-devel
```

6.3.9 Start the dev server

I just followed the steps in the Contributing guide

6.3.10 Run Streamlit

I just followed the steps in the Contributing guide and the servers are running.



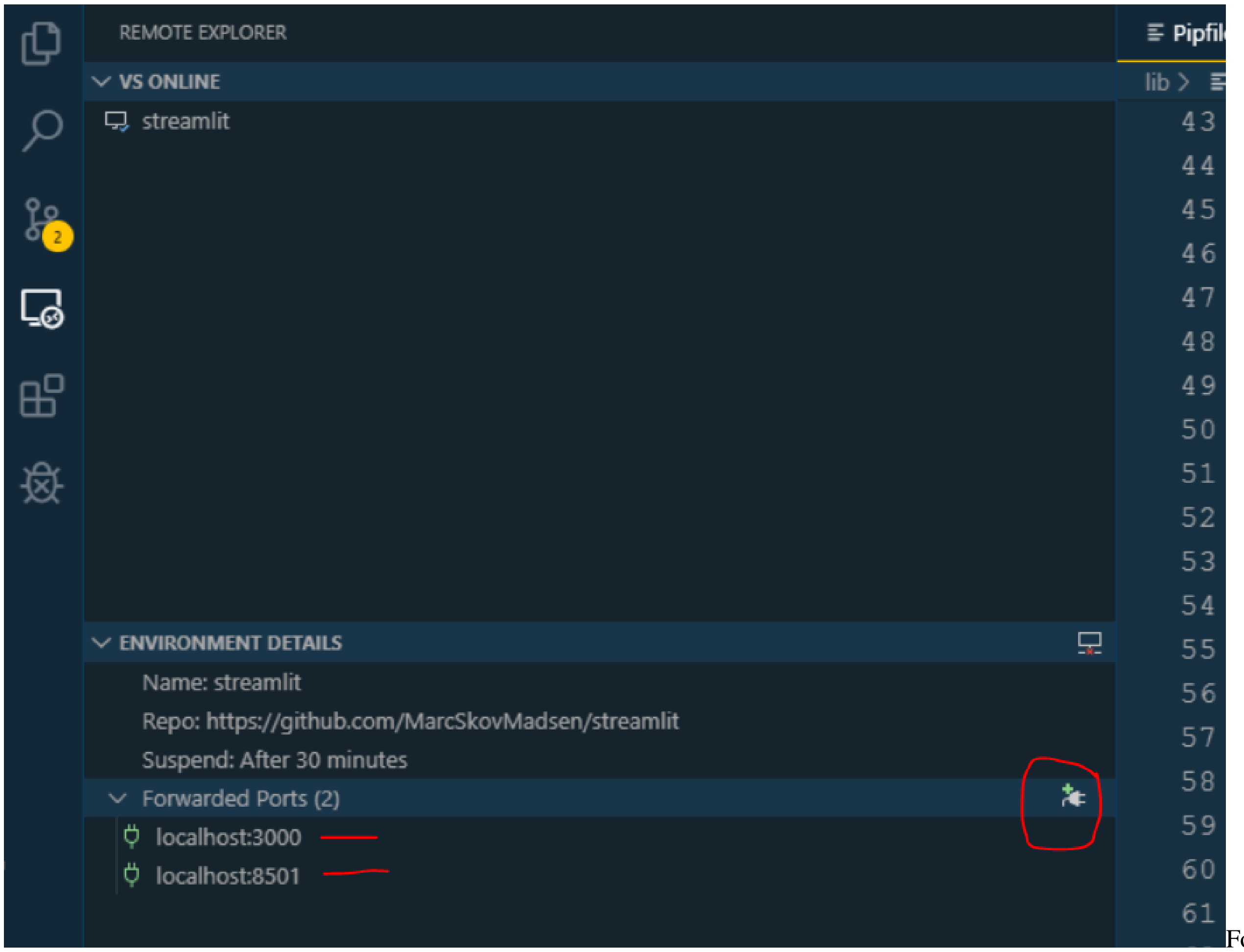
```
Starting the development server...

[hardsource:4afb8603] Writing new cache 4afb8603...
[hardsource:4afb8603] Tracking node dependencies with: yarn.lock.
=============

WARNING: You are currently running a version of TypeScript which is not offi
cially supported by typescript-estree.

You may find that it works just fine, or you may not.

SUPPORTED TYPESCRIPT VERSIONS: >=3.2.1 <3.6.0

YOUR TYPESCRIPT VERSION: 3.6.2

Please only submit bug reports when using the officially supported version.

=============
```

```
(lib) vsonline:~/workspace$ streamlit hello
generated new fontManager
2019-11-10T12:49:48Z.717 DEBUG   streamlit.logger: Initialized tornado logs

  Telemetry: As an open source project, we collect usage statistics.
  We cannot see and do not store information contained in Streamlit apps.

  If you'd like to opt out, add the following to ~/.streamlit/config.toml,
  creating that file if necessary:

    [browser]
    gatherUsageStats = false

Skipping PyPI version check
Setting up signal handler
Using selector: EpollSelector
Server state: None -> State.INITIAL
Starting server...
Serving static content from the Node dev server
Server started on port 8501
Creating new context for ws PREHEATED_REPORT_SESSION
No singleton. Registering one.
Watcher created for /home/vsonline/workspace/lib/streamlit/hello/hello.py
ReportSession initialized (id=0)
Beginning script thread
Server state: State.INITIAL -> State.WAITING_FOR_FIRST_BROWSER
Running script RerunData(widget_state=None)
OnScriptRunnerEvent: ScriptRunnerEvent.SCRIPT_STARTED
New browser connection: gather_usage_stats=True, sharing_enabled=False, max_
cached_message_age=2
OnScriptRunnerEvent: ScriptRunnerEvent.SCRIPT_STOPPED_WITH_SUCCESS
Starting new HTTP connection (1): checkip.amazonaws.com:80
OnScriptRunnerEvent: ScriptRunnerEvent.SHUTDOWN
http://checkip.amazonaws.com:80 "GET / HTTP/1.1" 200 13

  You can now view your Streamlit app in your browser.

  Network URL: http://172.17.0.2:3000
  External URL: http://168.63.5.173:3000

Watcher created for /home/vsonline/workspace/lib/streamlit/hello/__init__.py
Watcher created for /home/vsonline/workspace/lib/streamlit/hello/demos.py
```

Servers running

6.3.11 Forwarding the ports

In order to open Streamlit in your browser you need to forward the ports 3000 and 8501.



Name: streamlit

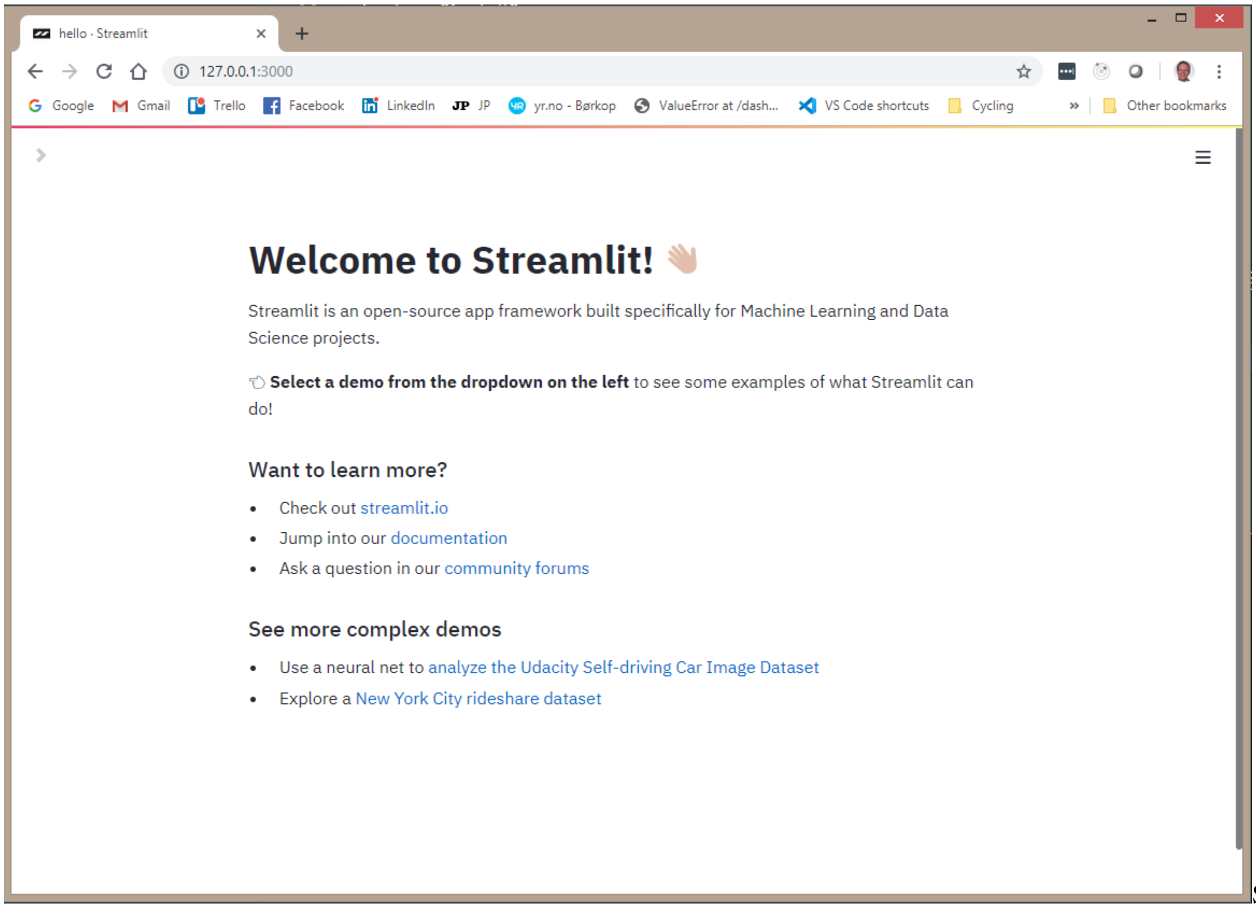
Repo: https://github.com/MarcSkovMadsen/streamlit

Suspend: After 30 minutes

Forward ports

6.3.12 Streamlit hello

And finally it works



Welcome to Streamlit! 👋

Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects.

👈 **Select a demo from the dropdown on the left** to see some examples of what Streamlit can do!

Want to learn more?

- Check out streamlit.io
- Jump into our documentation
- Ask a question in our community forums

See more complex demos

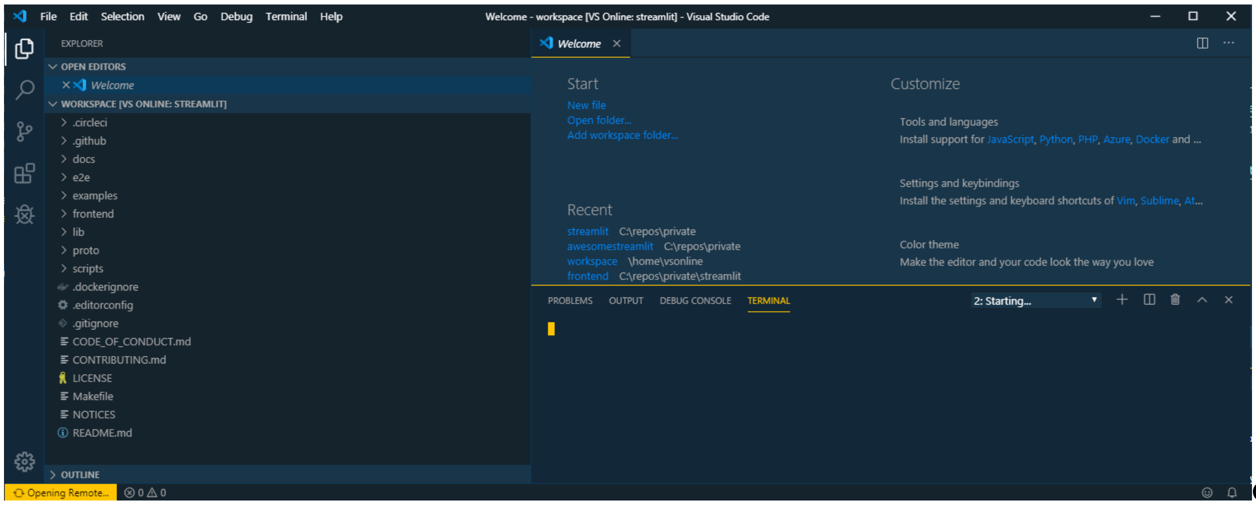
- Use a neural net to analyze the Udacity Self-driving Car Image Dataset
- Explore a New York City rideshare dataset

Streamlit hello

6.3.13 Troubleshooting

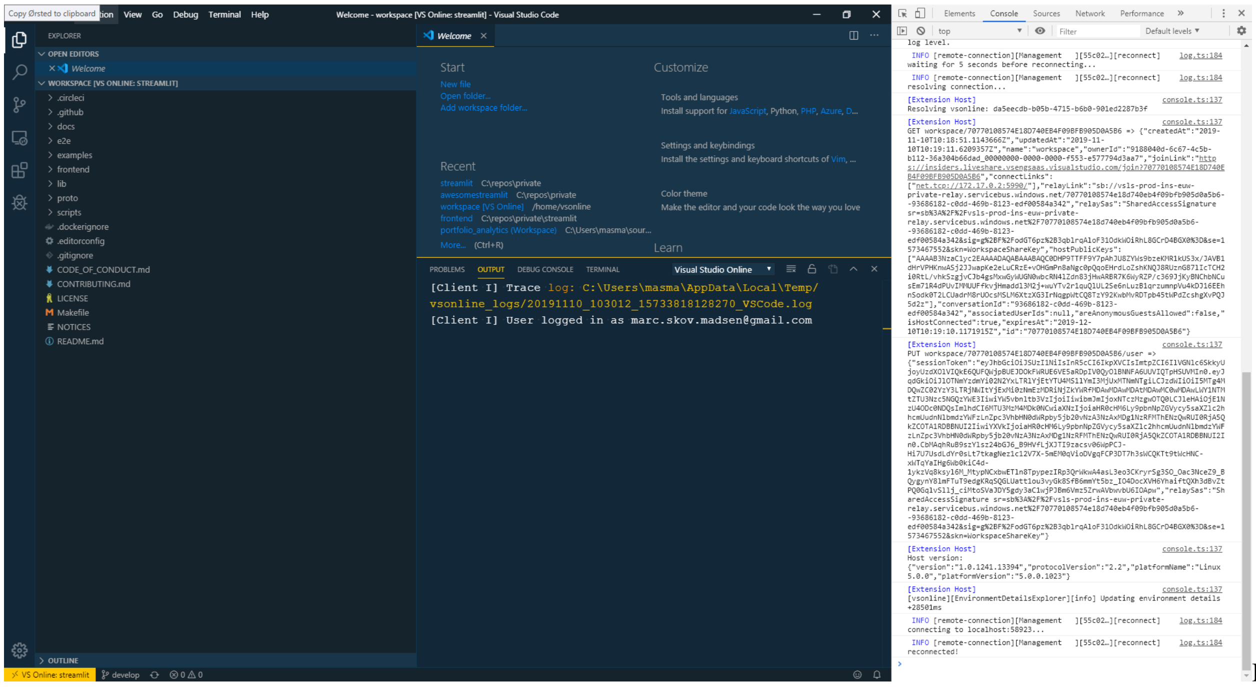
Connecting to the Remote Never Finishes

Sometimes VS Code cannot open the remote. I've filed an issue at MicrosoftDocs/vsonline issue #153



Opening Remote issue

I have experienced a few times that if I **Toggle Developer Tools** under the Help menu item then it starts reconnecting and succeeds. Strange :-)



```
[Client I] Trace log: C:\Users\masma\AppData\Local\Temp/
vsonline_logs/20191110_103012_15733818128270_VSCode.log
[Client I] User logged in as marc.skov.madsen@gmail.com
```

```
log level.
 INFO [remote-connection][Management   ][55c02…][reconnect]    log.ts:184
waiting for 5 seconds before reconnecting...
 INFO [remote-connection][Management   ][55c02…][reconnect]    log.ts:184
resolving connection...
[Extension Host]                                          console.ts:137
Resolving vsonline: da5eecdb-b05b-4715-b6b0-901ed2287b3f
[Extension Host]                                          console.ts:137
GET workspace/70770108574E18D740EB4F09BFB905D0A5B6 => {"createdAt":"2019-
11-10T10:18:51.1143666Z","updatedAt":"2019-11-
10T10:19:11.6209357Z","name":"workspace","ownerId":"9188040d-6c67-4c5b-
b112-36a304b66dad_00000000-0000-0000-f553-e577794d3aa7","joinLink":"http
s://insiders.liveshare.vsengsaas.visualstudio.com/join?70770108574E18D740E
B4F09BFB905D0A5B6","connectLinks":
["net.tcp://172.17.0.2:5990/"],"relayLink":"sb://vsls-prod-ins-euw-
private-relay.servicebus.windows.net/70770108574e18d740eb4f09bfb905d0a5b6-
-93686182-c0dd-469b-8123-edf00584a342","relaySas":"SharedAccessSignature
sr=sb%3A%2F%2Fvsls-prod-ins-euw-private-
relay.servicebus.windows.net%2F70770108574e18d740eb4f09bfb905d0a5b6-
-93686182-c0dd-469b-8123-
edf00584a342&sig=g%2BF%2FodGT6pz%2B3qblrqAloF31OdkWOiRhL8GCrD4BGX0%3D&se=1
573467552&skn=WorkspaceShareKey","hostPublicKeys":
["AAAAB3NzaC1yc2EAAAADAQABAAABAQC0DHP9TTFF9Y7pAhJU8ZYWs9bzeKMR1kUS3x/JAVB1
dHrVPHKnwASj2JJwapKe2eLuCRzE+vOHGmPn8aNgc0pQqoEHrdLoZshKNQJ8RUznG87lIcTCH2
i0RtL/vhkSzgjvCJb4gsMxwGyWUGN0wbcRN4lZdn83jHwARBR7K6WyRZP/c369JjKyBNChbNCu
sEm71R4dPUvIMMUUFfkvjHmaddl3M2j+wuYTv2r1quQ1UL2Se6nLuzB1qrzumnpVu4kDJ16EEh
nSodk0T2LCUadrM8rUOcsMSLM6XtzXG3IrNqgpWtCQ8TzY92KwbMvRDTpb45tWPdZcshgXvPQJ
5d2z"],"conversationId":"93686182-c0dd-469b-8123-
edf00584a342","associatedUserIds":null,"areAnonymousGuestsAllowed":false,"
isHostConnected":true,"expiresAt":"2019-12-
10T10:19:10.1171915Z","id":"70770108574E18D740EB4F09BFB905D0A5B6"}
[Extension Host]                                          console.ts:137
PUT workspace/70770108574E18D740EB4F09BFB905D0A5B6/user =>
{"sessionToken":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IlVGNlc6SkkyU
joyUzdXOlVIQkE6QUFQWjpBUEJDOkFWRUE6VE5aRDpIV0QyOlBNNFA6UUVIQTpHSUVMIn0.eyJ
qdGkiOiJlOTNmYzdmYi02N2YxLTRlYjEtYTU4MS1lYmI3MjUxMTNmNTgiLCJzdWIiOiI5MTg4M
DQwZC02YzY3LTRjNWItYjExMi0zNmEzMDRiNjZkYWRfMDAwMDAwMDAtMDAwMC0wMDAwLWY1NTM
tZTU3Nzc5NGQzYWE3IiwiYW5vbnltb3VzIjoiIiwibmJmIjoxNTczMzgwOTQ0LCJleHAiOjE1N
zU4ODc0NDQsImlhdCI6MTU3MzM4MDk0NCwiaXNzIjoiaHR0cHM6Ly9pbnNpZGVycy5saXZlc2h
hcmUudnNlbmdzYWFzLnZpc3VhbHN0dWRpby5jb20vNzA3NzAxMDg1NzRFMThENzQwRUI0RjA5Q
kZCOTA1RDBBNUI2IiwiYXVkIjoiaHR0cHM6Ly9pbnNpZGVycy5saXZlc2hhcmUudnNlbmdzYWF
zLnZpc3VhbHN0dWRpby5jb20vNzA3NzAxMDg1NzRFMThENzQwRUI0RjA5QkZCOTA1RDBBNUI2I
n0.CbMAqhRuB9szYlsz24bGJ6_B9HVfLjXJTI9zacsv06WpPCJ-
Hi7U7UsdLdYr0sLt7tkagNez1c12V7X-5mEM0qVioDVgqFCP3DT7h3sWCQKTt9tWcHNC-
xWTqYaIHg6Wb0kiC4d-
1ykzVq8ksyl6M_MtypNCxbwETln8TpypezIRp3QrWkwA4asL3eo3CKryrSg3SO_Oac3NceZ9_B
QygynY8lmFTuT9edgKRqSQGLUatt1ou3vyGk8SfB6mmYt5bz_IO4DocXVH6YhaiftQXh3dBvZt
PQ0GqlvSllj_ciMtoSVaJDY5gdy3aC1wjPJBm6Vmz5ZrwAVbwvbU6IOApw","relaySas":"Sh
aredAccessSignature sr=sb%3A%2F%2Fvsls-prod-ins-euw-private-
relay.servicebus.windows.net%2F70770108574e18d740eb4f09bfb905d0a5b6-
-93686182-c0dd-469b-8123-
edf00584a342&sig=g%2BF%2FodGT6pz%2B3qblrqAloF31OdkWOiRhL8GCrD4BGX0%3D&se=1
573467552&skn=WorkspaceShareKey"}
[Extension Host]                                          console.ts:137
Host version:
{"version":"1.0.1241.13394","protocolVersion":"2.2","platformName":"Linux
5.0.0","platformVersion":"5.0.0.1023"}
[Extension Host]                                          console.ts:137
[vsonline][EnvironmentDetailsExplorer][info] Updating environment details
+28501ms
 INFO [remote-connection][Management   ][55c02…][reconnect]    log.ts:184
connecting to localhost:58923...
 INFO [remote-connection][Management   ][55c02…][reconnect]    log.ts:184
reconnected!
```

Developer Tools

The Contributing Guide Does Not Work for me

I could not exactly follow the Contributing guide. I've filed the issues I saw as streamlit/streamlit issue #665.

Build Python3.7.4 from scratch

At some stage i could not get pyenv to work. So I followed the How to Install Python 3.7 on Debian 9 Guide with

- `3.7.3` replaced by `3.7.4`.
- `make -j 8` replaced by `make -j 4`

Please note that there will be **plenty of time for coffea** as the installation and tests take 30 minutes!

After verifying the installation I removed the temporary `Python-3.7.4.tar.xz` file and `Python-3.7.4` folder.

```
rm Python-3.7.4.tar.xz
sudo rm -rf Python-3.7.4
```

Package python3-distutils is not Available

Maybe because of building 3.7.4 from scratch I ran into

```
vsonline:~/workspace$ sudo apt install python3-distutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package python3-distutils is not available, but is referred to by another package.
```

(continues on next page)

(continued from previous page)

```
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
  libpython3.7-stdlib

E: Package 'python3-distutils' has no installation candidate
vsonline:~/workspace$ sudo apt install libpython3.7-stdlib
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package libpython3.7-stdlib is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
  idle-python3.7 python3-tk python3-lib2to3 python3-distutils

E: Package 'libpython3.7-stdlib' has no installation candidate
vsonline:~/workspace$ sudo apt install idle-python3.7 python3-tk python3-lib2to3␣
↪python3-distutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package python3-distutils is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
  libpython3.7-stdlib

Package python3-lib2to3 is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

Package idle-python3.7 is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'idle-python3.7' has no installation candidate
E: Package 'python3-lib2to3' has no installation candidate
E: Package 'python3-distutils' has no installation candidate
```

Makefile:46: recipe for target 'pipenv-install' failed

When running

```
make all-devel
```

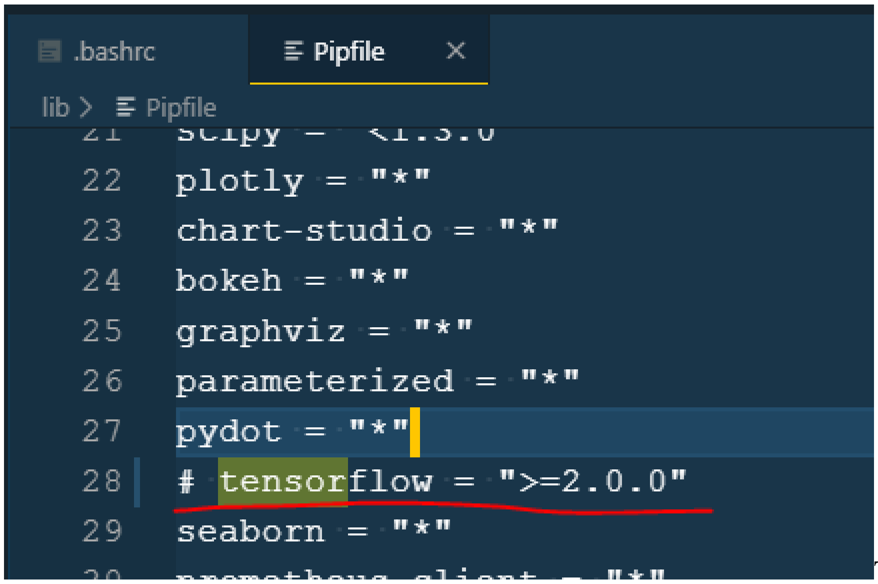
I get the below error.

```
pip/_internal/utils/misc.py", line 703, in call_subprocess
    raise InstallationError(
pipenv.patched.notpip._internal.exceptions.InstallationError: Command "python setup.
↪py egg_info" failed with error code 1 in /tmp/tmp5wxfmdotbuild/functools32/
Makefile:46: recipe for target 'pipenv-install' failed
make: *** [pipenv-install] Error 1
```

The root cause is Tensorflow. Tensorflow does not support installation with pipenv according to pyp/pipenv issue #2619 and similar issues.

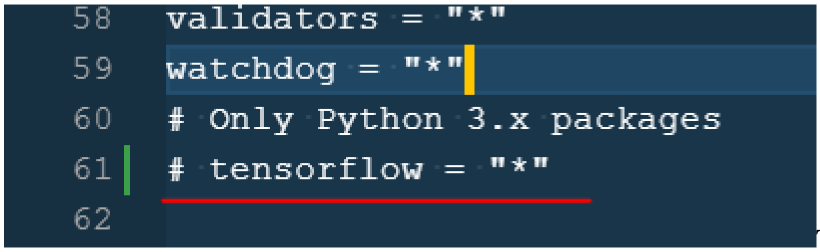
The solution is to remove it from the Piplock File and install it manually

```
pip install tensorflow>=2.0.0
```



```
plotly = "*"
chart-studio = "*"
bokeh = "*"
graphviz = "*"
parameterized = "*"
pydot = "*"
# tensorflow = ">=2.0.0"
seaborn = "*"
```

Tensorflow removed from pipenv file



```
validators = "*"
watchdog = "*"
# Only Python 3.x packages
# tensorflow = "*"
```

Tensorflow removed from pipenv file

CHAPTER 7

Indices and tables

- genindex
- modindex
- search