

Advanced Python

Class 6

Pepe Bonet Giner

19th January 2023

Index Class 6

Topic 1: Recap Class 5

Topic 2: Repository Structure

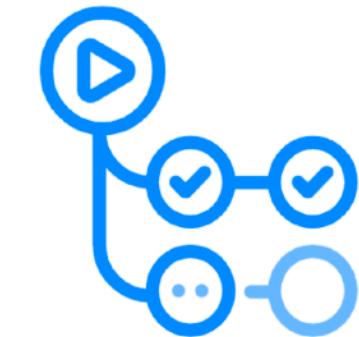
Topic 3: Final Project

Topic 4: Final Words

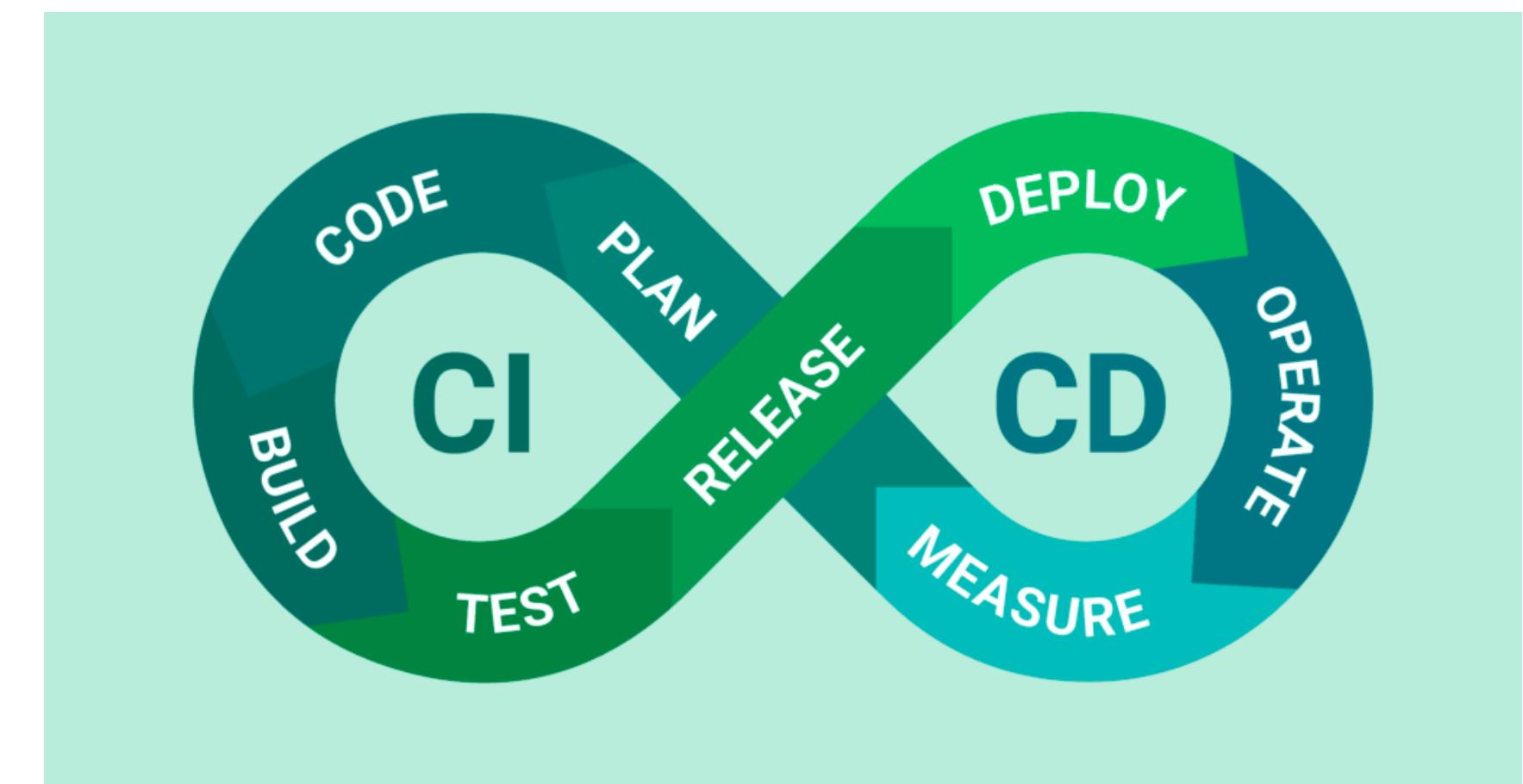
Topic 1: Recap Class 5

What is CI / CD ?

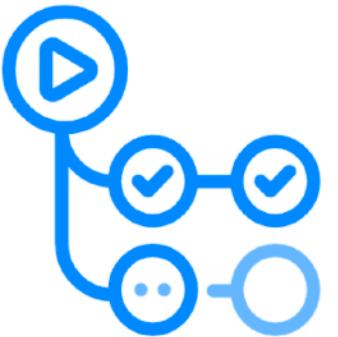
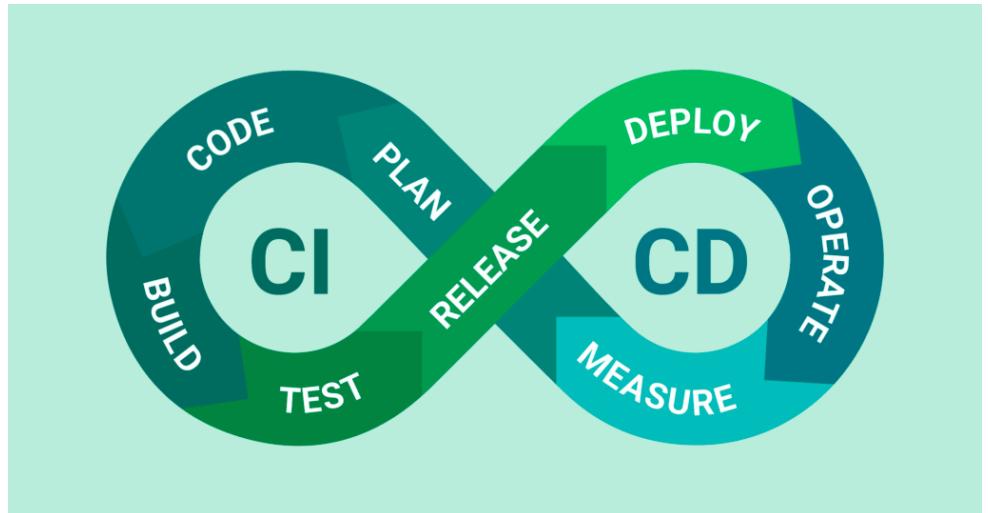
- **Continuous Improvement (CI)** refers to the practice of regularly integrating code changes from multiple developers into a shared repository.
 - The goal is to detect issues and conflicts early in the development cycle and ensure that the codebase remains stable.
- **Continuous Deployment (CD)** focuses on automating the deployment of applications to various environments (e.g., staging, production) after passing the CI stage.
 - It involves automating the steps required to build, test, and deploy software, reducing manual effort and ensuring consistent and reliable deployments.



GitHub Actions



What are the benefits?



GitHub Actions



- **Faster Time to Market:** Automating processes reduces manual effort and speeds up the delivery of software updates.
- **Improved Code Quality:** Continuous integration and automated testing catch errors early, ensuring code quality and reducing the risk of introducing bugs.
- **Collaboration and Feedback:** Code reviews and feedback from peers help improve code quality and foster collaboration among team members.
- **Reliable Deployments (Less human mistakes):** Automating the deployment process minimizes the risk of configuration errors and ensures consistent deployments across environments.

Makefile

- Makefile is a special file used to automate the process of compiling and building software by using “make”
- Every blue box is a rule
- Every rule has a target (a name) (Green box)
- Every name can have dependencies (pink)
- Every rule and name has a command to be run (orange boxes)

```
install: pip install --upgrade pip && pip install -r requirements.txt
pylint: pylint *
flake8: flake8 *
test: python -m unittest
all: install lint test
```

What is Multiprocessing in Python

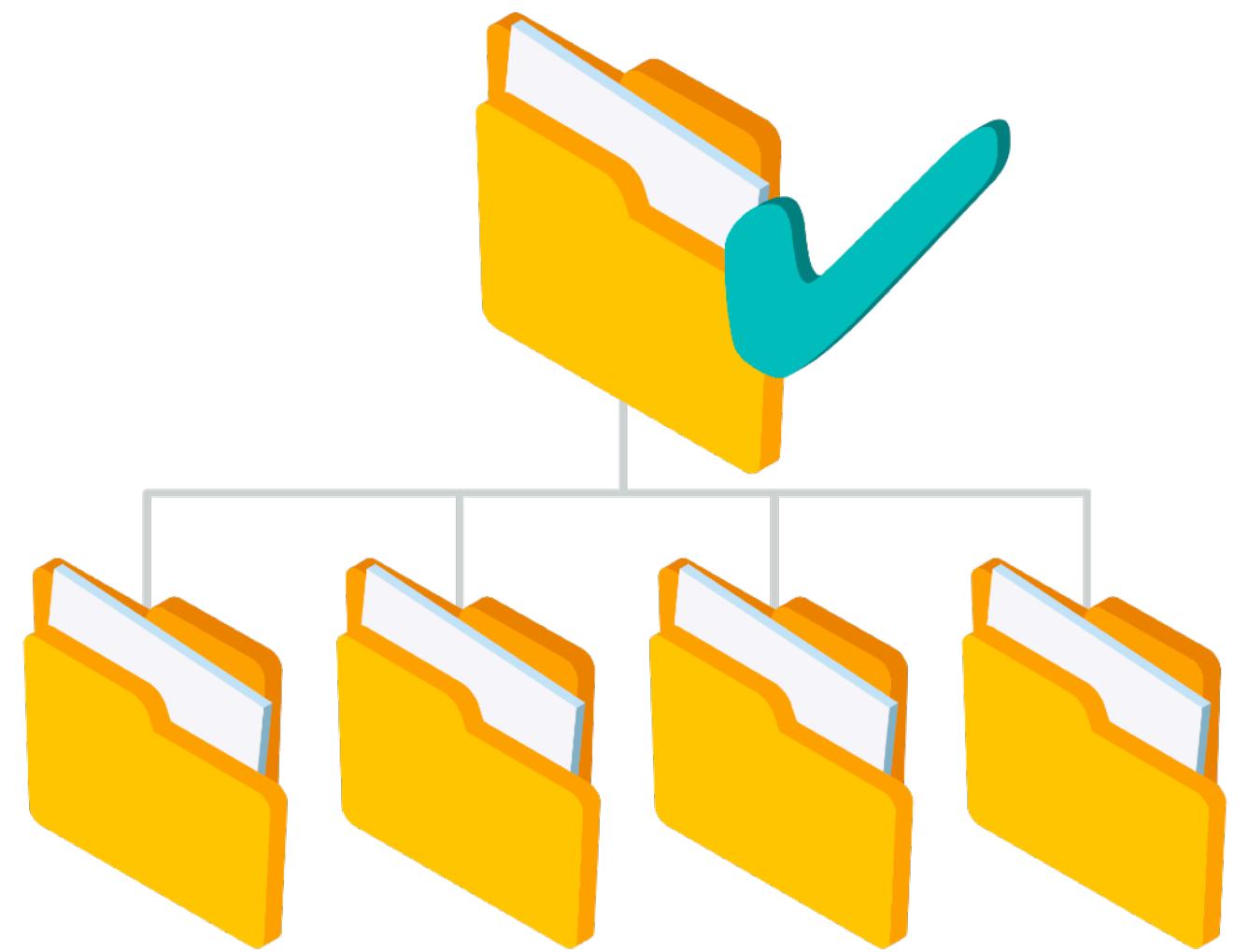
Essentially is a tool to parallelize tasks to take advantage of multiple CPUs to reduce computation time

```
@click.option(  
    '-cpu', '--cpus', default=1, help='number of processes to be used, default 1'  
)  
@click.option(  
    '-o', '--output', default='', help='Output file'  
)  
def preprocess(**kwargs):  
    """Preprocess data for training"""  
  
    args = Namespace(**kwargs)  
  
    if args.split_features:  
        split_preprocess(  
            args.features, args.output, args.save_tsv, args.cpus,  
            args.split_type, args.positions, args.feature_type  
        )  
    else:  
        no_split_preprocess(  
            args.features, args.output, args.cpus, args.feature_type  
        )
```

```
print('Splitting original file...')  
os.mkdir(tmp_folder);  
os.mkdir(tmp_train); os.mkdir(tmp_test); os.mkdir(tmp_val)  
cmd = 'split -l {} {} {}'.format(2000, features, tmp_folder)  
subprocess.call(cmd, shell=True)  
  
if positions:  
    print('Getting position file...')  
    positions = pd.read_csv(positions, sep='\t')  
  
    print('Extracting features to h5 and tsv files...')  
    counter = 0  
  
    f = functools.partial(split_sets_files, tmp_folder=tmp_folder, \  
                          counter=counter, tsv_flag=tsv_flag, output=output, \  
                          tmps=os.path.dirname(features), split_type=split_type, \  
                          positions=positions, feature_type=feature_type)  
    with Pool(cpus) as p:  
        for i, rval in enumerate(p.imap_unordered(f, os.listdir(tmp_folder))):  
            counter += 1
```

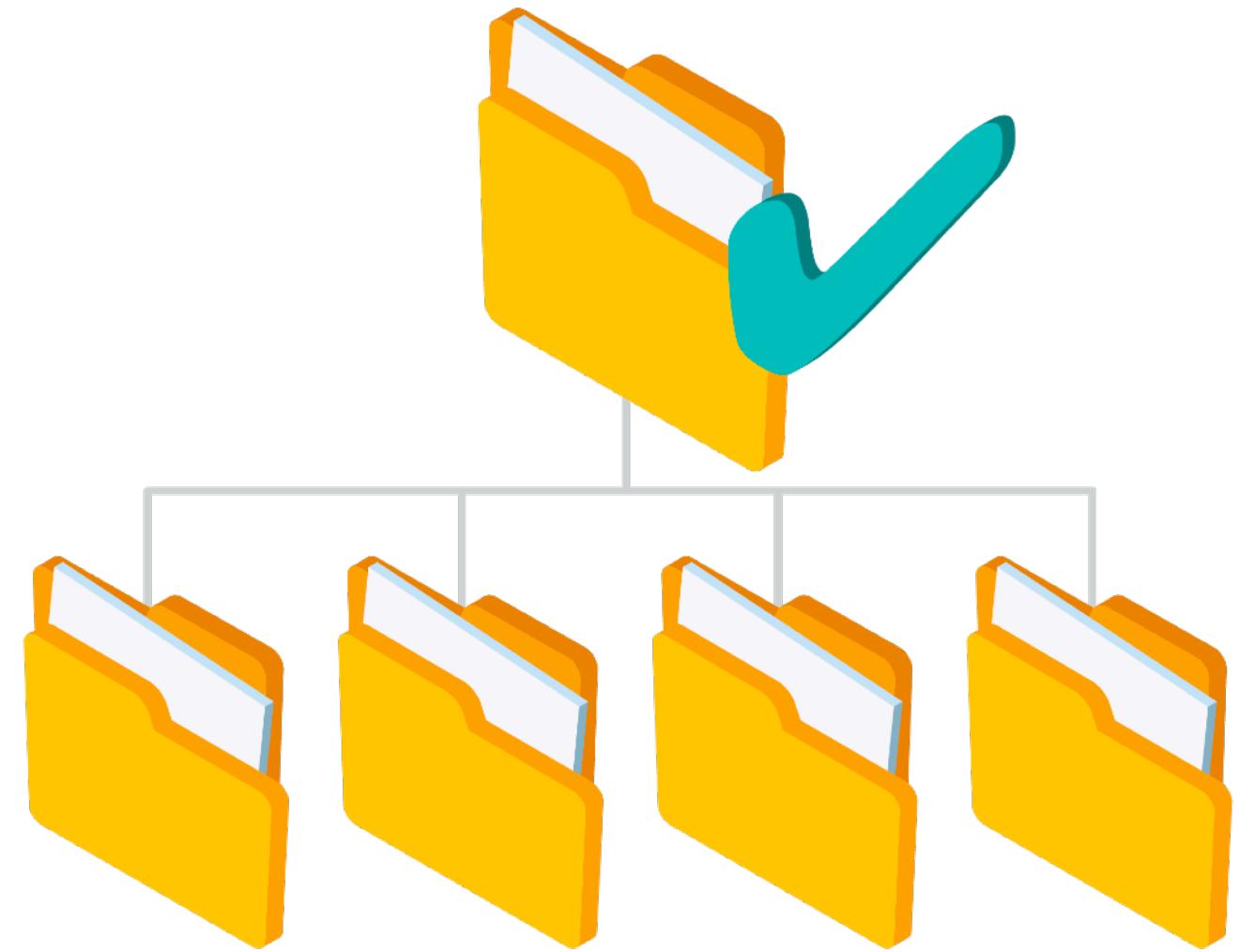
Topic 2: Repository Structure

What do we need in our repository? (File-wise)

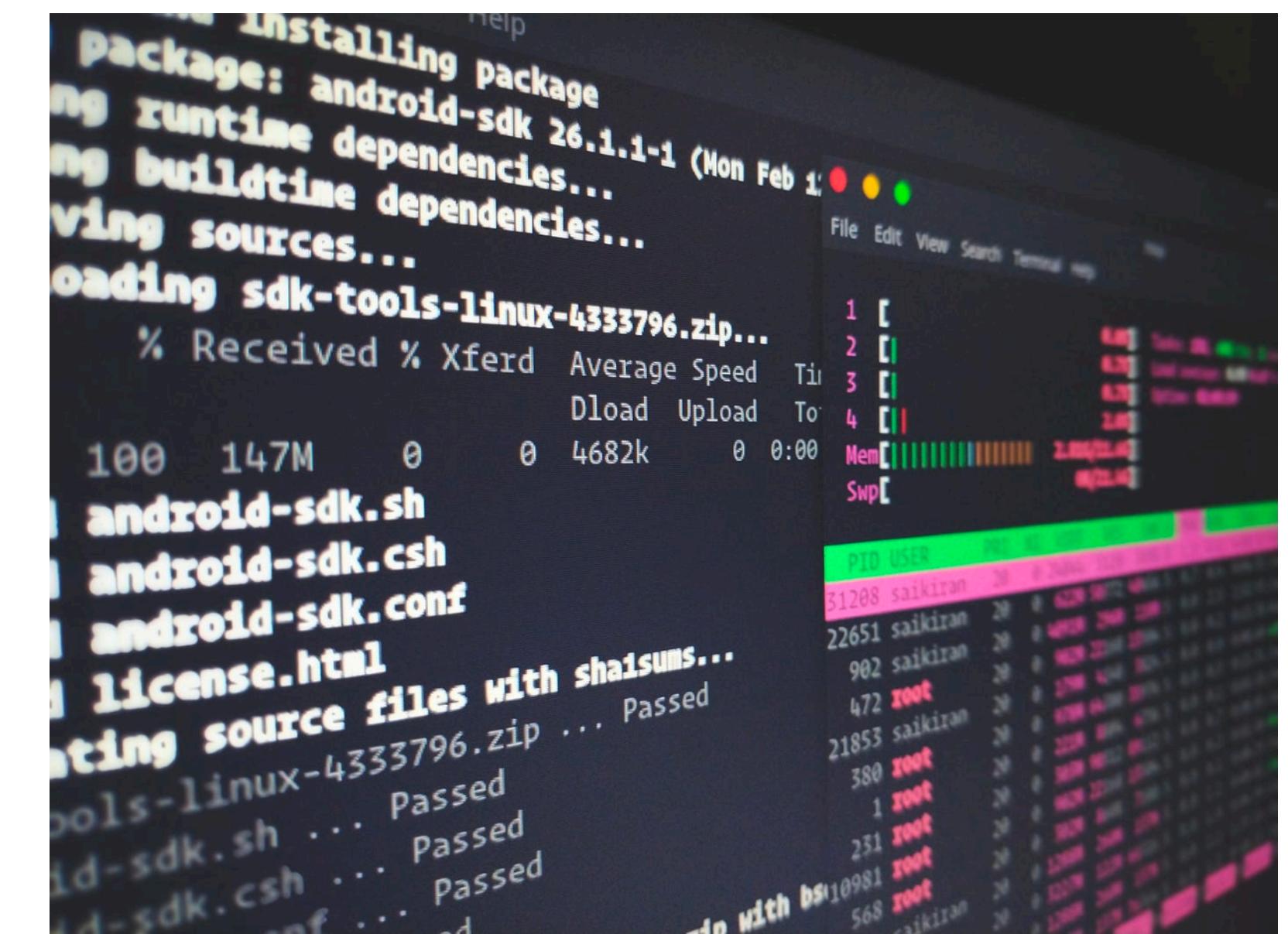


What do we need in our repository? (File-wise)

- requirements.txt
- Scripts folder
- Datasets folder
- Tests folder
- .gitignore
- .github/workflows/checks.yml
- Makefile
- .flake8
- .pylintrc



What commands we will need for sure?



What commands we will need for sure?

- Command for installing the dependencies in the environment
 - Command for testing code
 - Command for linting code
 - Commands for running the code

```
Installing package
Package: android-sdk 26.1.1-1 (Mon Feb 12 10:00:00 UTC 2018)
    runtime dependencies...
    buildtime dependencies...
    sources...
Loading sdk-tools-linux-4333796.zip...
% Received % Xferd  Average Speed  Time: 0:00:00
                                         Dload  Upload Total   Spent
100 147M     0      0  4682k      0  0:00
android-sdk.sh
android-sdk.csh
android-sdk.conf
license.html
Comparing source files with sha1sums...
ools-linux-4333796.zip ... Passed
id-sdk.sh ... Passed
id-sdk.csh ... Passed
id-sdk.conf ... Passed
Comparing source files with sha1sums...
ools-linux-4333796.zip ... Passed
id-sdk.sh ... Passed
id-sdk.csh ... Passed
id-sdk.conf ... Passed
Comparing source files with sha1sums...
ools-linux-4333796.zip ... Passed
id-sdk.sh ... Passed
id-sdk.csh ... Passed
id-sdk.conf ... Passed
```

We are missing something. Any ideas?

We are missing something. Document what we have done!

How do we tell people how to run our code? Or what the code is about?

- We do it through the README.md file
- This file is in every single repository and is essential to understand what your code does
- It's a Markdown file (You can do similar things as in JN)

pandas: powerful Python data analysis toolkit

Testing	Unit Tests failing codecov 96%
Package	pypi v2.1.4 PyPI downloads 149M/month Anaconda.org 2.2.0rc0 Conda downloads 47M
Meta	powered by NumFOCUS DOI 10.5281/zenodo.3509134 license BSD join Slack information

What is it?

pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.

Table of Contents

- [Main Features](#)
- [Where to get it](#)
- [Dependencies](#)
- [Installation from sources](#)
- [License](#)

Follow along

- Let's build our readme and start understanding a bit about what goes in there:
 - Project and repo explanation
 - Commands install and run the code
 - Explanation of the commands
 - Explanation of the CLI Arguments to a certain level

Topic 3: Final Project

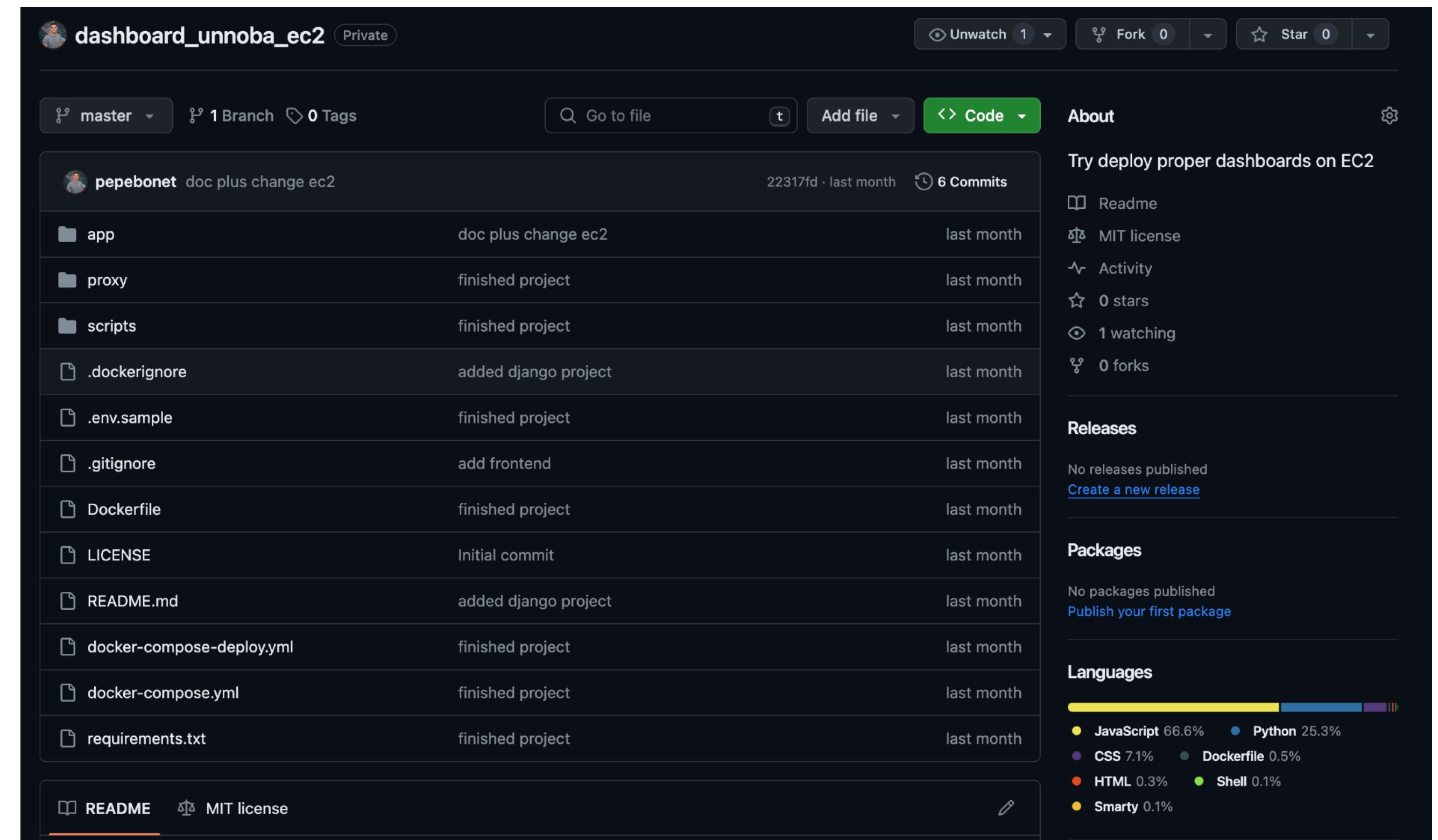
Final Project Objective

- The goal is to apply everything we have learned to a data analysis / EDA / AI task on a dataset
- You can prepare the dataset for ML models and even try some if you want to as well

Final Project Requisites

A GitHub repository containing all the things we see during the course for a given dataset:

- Scripting (VStudio)
- Classes
- Click
- Debugging
- GitHub 1 (Version Control)
- GitHub 2 (CD & CI)
- Project Structure
- Testing
- Linting + Clean Code



Final Project Requisites

- A requirements.txt file with the dependencies and their versions
- At least 2 Different scripts with functions/Classes
- At least I want to see two classes
- One of the scripts at least needs to be using click and should be able to be called using arguments from the command line
- The main script needs to call at least a second script.
- I have to see different commits and that their names correspond with the changes you did to the code
- At least 10 tests (either use pylint or unittest)
- Makefile to install dependencies, run tests, lint, and even run code if you want
- Usage of at least one linting tool
- Usage of try and except statements
- Well-documented project in the README.md
- GitHub Actions for automated testing (and Linting if possible)

I should be able to follow the steps in your README and make the code work

It's on you how much you want to work on it, but...

I think I am giving you the chance of in two weeks having something in your CV. Maybe it can be nice to have something like this (right image) if it is something people ask for when you are looking for a job

PROJECTS

Auction Final Price Prediction

Tech Stack: Python, CatBoost, SQL

Description: A machine learning model that can predict the final price the match worn shirts in auctions.

- designed the model that first classifies a product into high price or low price product, then use two different models to further predict the precise final price to mitigate the problem of highly skewed product price distribution

Topic 4: Final Words

Keep Learning!

Job (Projects)

University
Courses

Videos

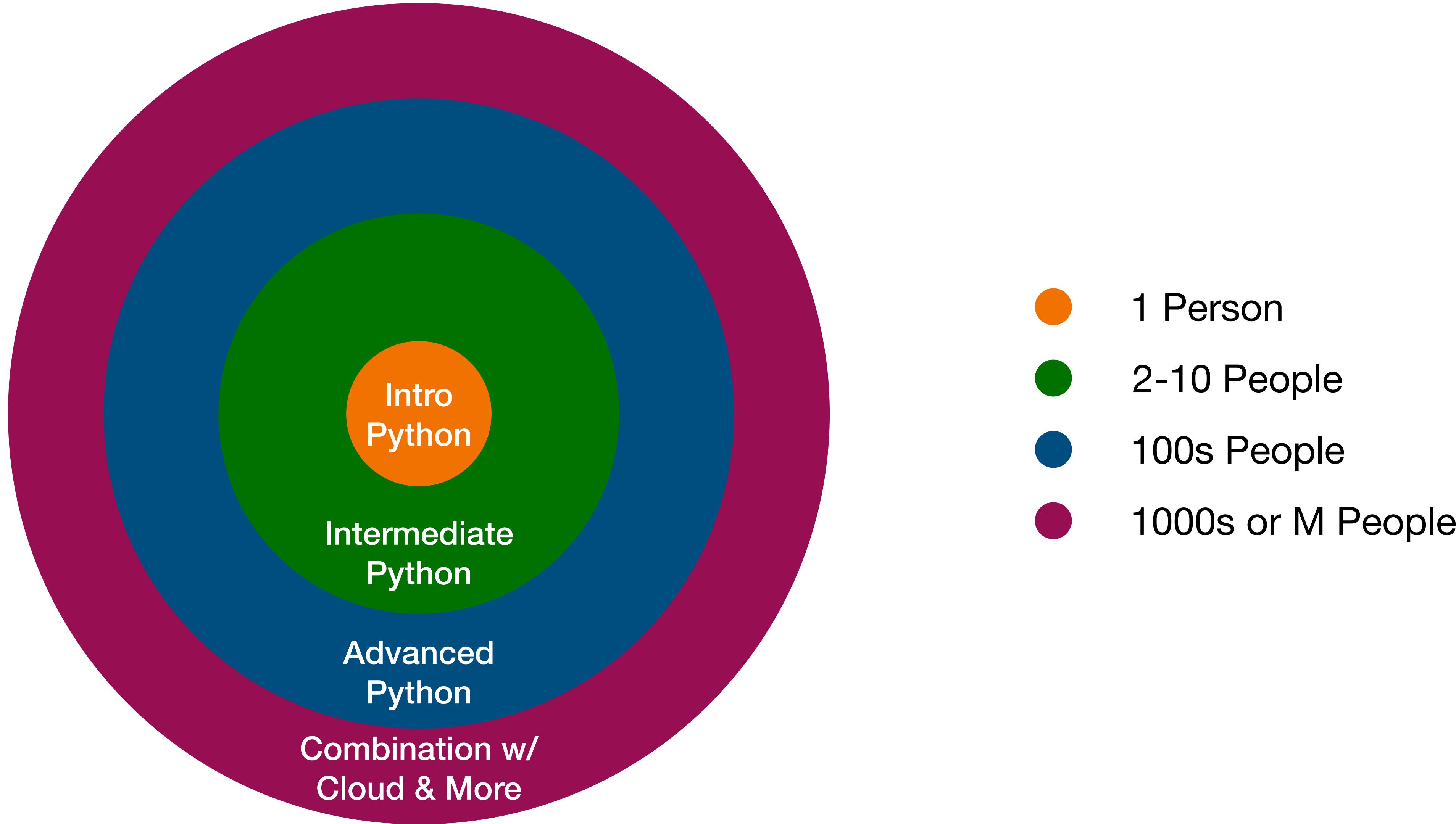
Own Projects

Practice

Books

Online
Courses

What is the last layer about?



What is the last layer about?



1000s or M People

Understanding and
good usage of the cloud

Combination with other
programming languages
(JS, HTML, CSS, SQL)

Containerize
frameworks (Docker,
Kubernetes, Terraform)

Security

Web Development

Code efficiency

But let's make something clear: That's my view

I have an odd perspective where I either do all of that or I cannot develop a product that generates value and (hopefully) revenue

Ideally you have a team and you do the thing on the right. Why? Because unless you are crazy good (I am not there. At least yet), doing it alone can be costly.

Another perspective is: Specialize!
(Probably better to find a job)

Frontend

Cloud Engineer

Backend

Data Engineer

Data Science

ML Engineer

Full Stack

Data Analyst

Many more...

We learned a lot in less than two weeks

- Scripting (VStudio)
- Classes
- Click
- Debugging
- GitHub 1 (Version Control)
- GitHub 2 (CD & CI)
- Project Structure
- Testing
- Multiprocessing
- Exceptions + assert
- Linting
- Makefile

Thank you!

Feel Free to contact me!

- Linkedin: Jose Bonet Giner
- Personal mail: pepebogi5@gmail.com
- Company website: hyntsanalytics.com
- Blog: pepesjourney.com

The logo consists of the word "hynts" in a lowercase sans-serif font. The letter "i" has a small blue dot above it.

Thank you! / Last exercise ;)

Fill in the survey / Feedback form!