Typing in Python

paris.py - 18/12/18

About me

- *Lead Developer @ Wiremind (we're hiring ! ;))*
- *Main development languages:*
  - *Python 2 & 3*
  - *Typescript*

Why typing ?

"Now! *That* should clear up a few things around here!"

Dynamic typing is Great !

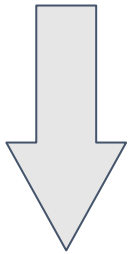
- Flexible
- Powerful
- Easy

But sometimes you need static typing

- code is easier to read
- Machine-checked documentation
- makes IDEs smarter
- helps find bugs

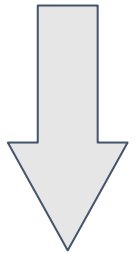
Why typing ?

Dynamic typing



Perfect for prototyping & small projects

Static typing



Perfect for complex application logic & projects that need to be maintained

Typing in Python

- Typing is builtin in Python 3

```python
from typing import Dict

def f(x: Dict[str, int]) -> int:
        # python >= 3.6 syntax
    y: str = "x"
    z = "x"  # type: str
    return 1
```

- But is available as an external package in Python 2 !

```python
from typing import Dict

def f(x):
    # type: (Dict[str, int]) -> int
    z = "x"  # type: str
    return 1
```

Typing in Python

- No type checking at runtime
- You need to use a type-checking tool: mypy
  - pip install mypy
  - mypy yourproject/ (mypy --py2 yourproject/ for python2)
- Things to know about mypy:
  - A lot of options ! (https://mypy.readthedocs.io/en/latest/command_line.html)
  - Uses library stubs
    - typeshed
    - custom ones (https://github.com/dropbox/sqlalchemy-stubs/tree/master/sqlalchemy-stubs for example)
  - Infers types dynamically:
    - x = 1 # infers int
    - x = {} # will ask you for help
  - By default does not typecheck untyped functions

Typing in Python

- Static typing is entirely optional
- Static typing handles Python duck-typing (Mapping, Iterable, Protocols, etc.)

```
from typing import Dict

def f(x: Iterable[int]): ...

f([1, 2])  # OK
f({2, 3, 4})  # OK
f(1)  # NOT OK
```

- Allows for gradually typing a codebase
- Supports a lots of python features and idioms

```
from typing import TypeVar, List

T = TypeVar("T", int, A)

def f(x):
  # type: (T) -> List[T]
  return [x]

f(1)  # type: List[int]
f(A())  # type: List[A]
f("X")  # NO
```

```
T = TypeVar('T')

class Stack(Generic[T]):
    def __init__(self) -> None:
        # Create an empty list with items of type T
        self.items: List[T] = []
    def push(self, item: T) -> None:
        self.items.append(item)
    def pop(self) -> T:
        return self.items.pop()

stack = Stack()
stack.push(2)  # ok
stack.pop()  # int
stack.push('x') # error
```

Typing an existing project

01

Warm up

Read mypy documentation & cheatsheets

~1 hour

.

02

Get mypy running without errors

mypy
--ignore-missing-imports
yourprj/

~2-3 hours

.

03

Check untyped functions

mypy
--ignore-missing-imports
--check-untyped-defs
yourprj/

~1 day

.

04

Add it to you CI

You have a minimal setup. But to take advantage of mypy, you'll want to start typing functions

.

05

Annotate you core modules

Once a module is fully annotating, set disallow_untyped_defs = True for that module in mypy.ini

~ 2-3 days

.

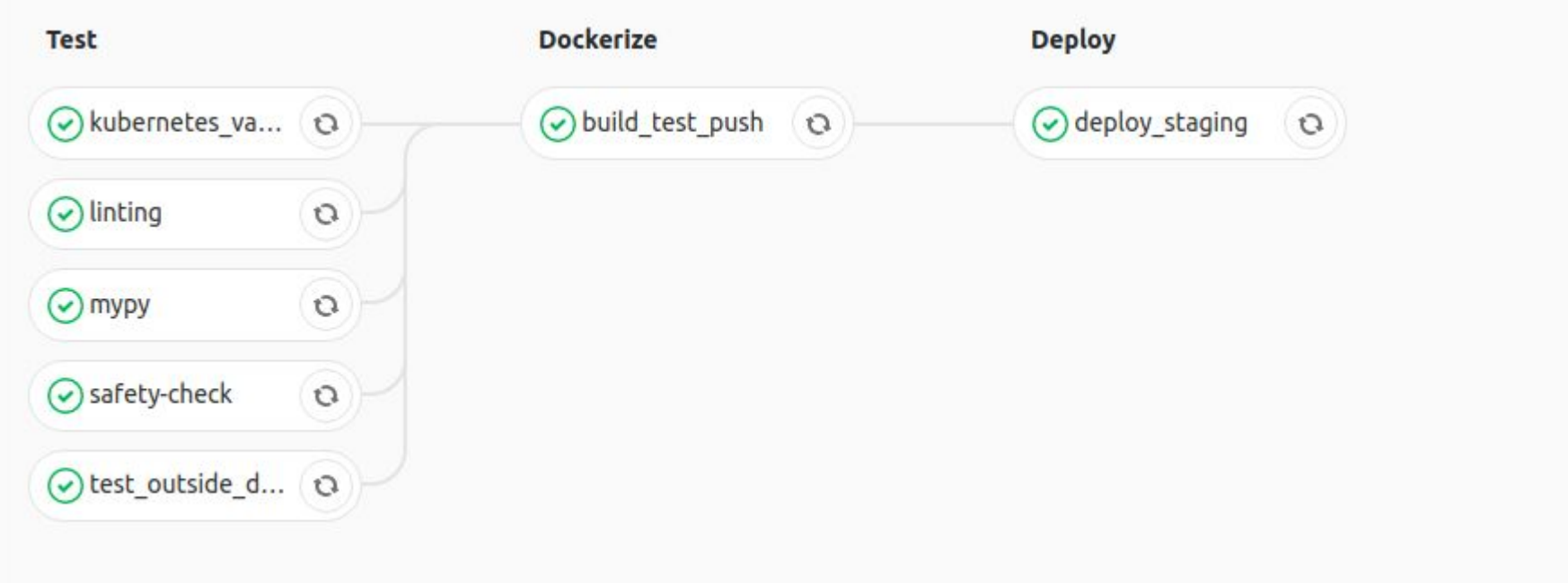
06

Annotate other modules...

And Enjoy :)

.

Typing an existing project



Questions ?