Continuous Performance Analysis

for Python with

{codspeed}

whoami

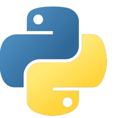
Arthur Pastel
Software Engineer

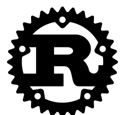
twitter.com/art049

github.com/art049

linkedin.com/in/arthurpastel

Weapons of choice:





Open Source Developer:

Built ODMantic 🍃

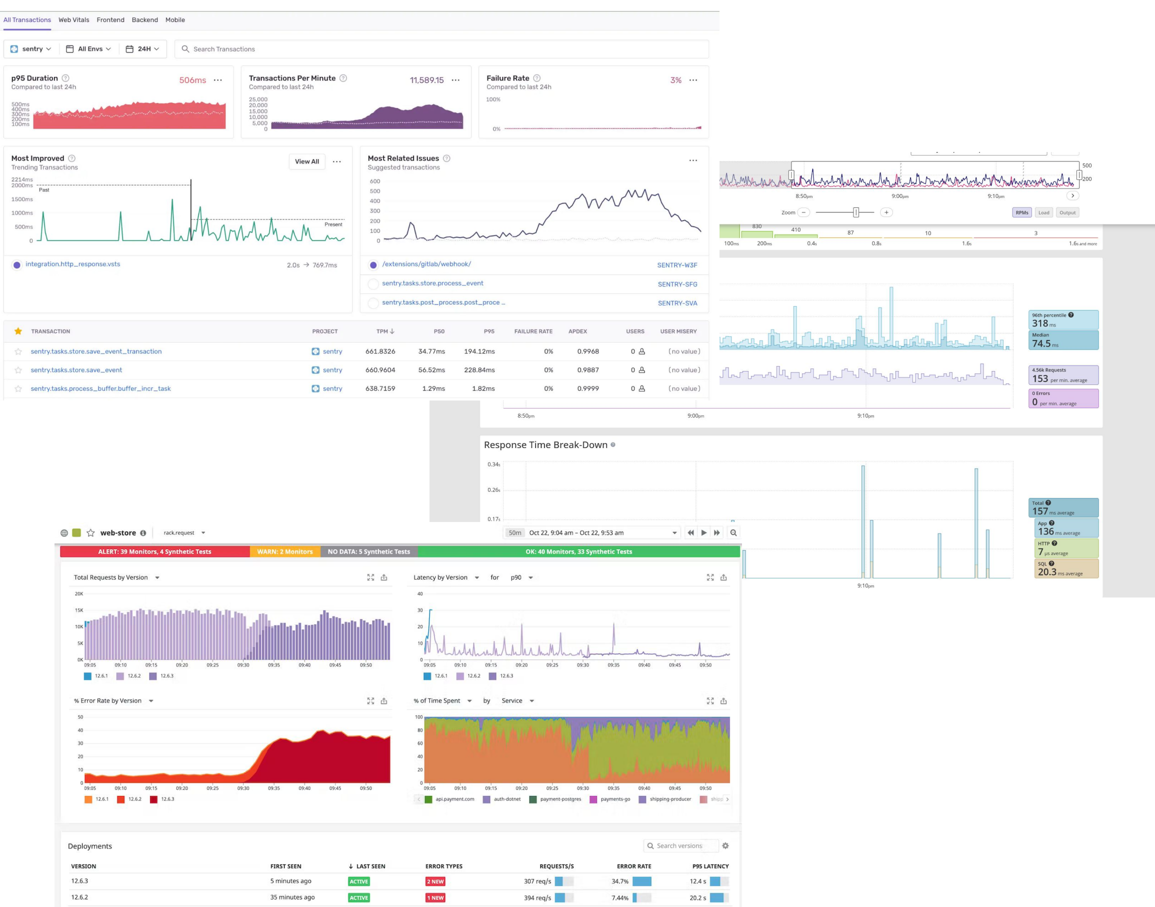
Building

Software Performance?

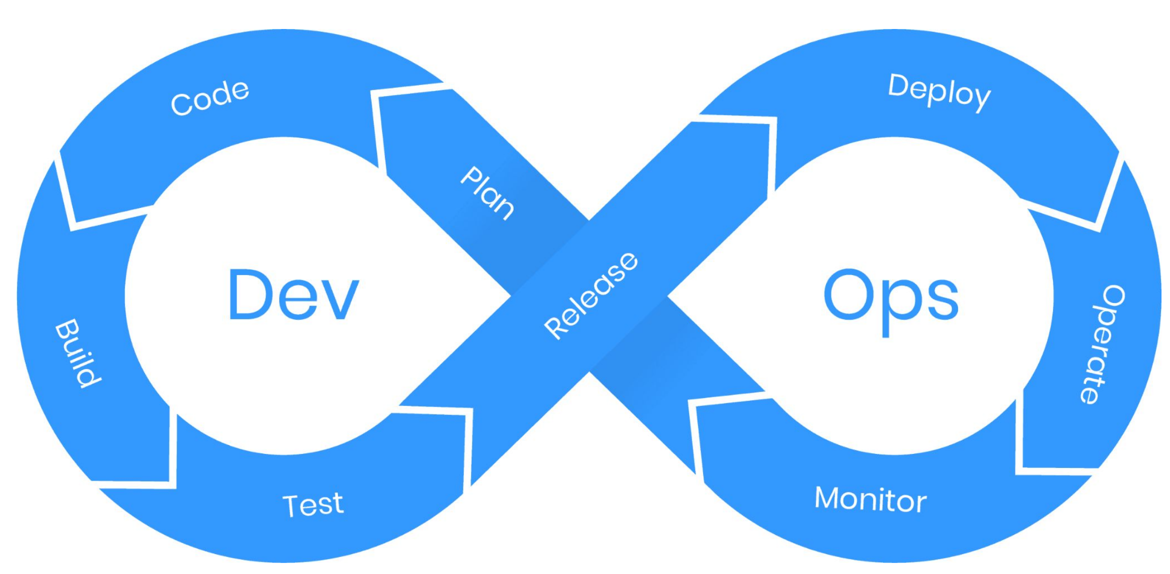
- Execution speed
- Throughput

APMs

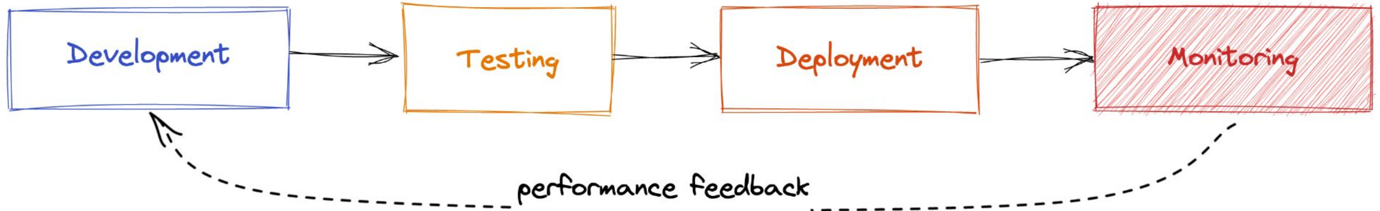
- Sentry
- Datadog
- Blackfire
- Cloud Providers



The Software Development Life Cycle(aka SDLC)



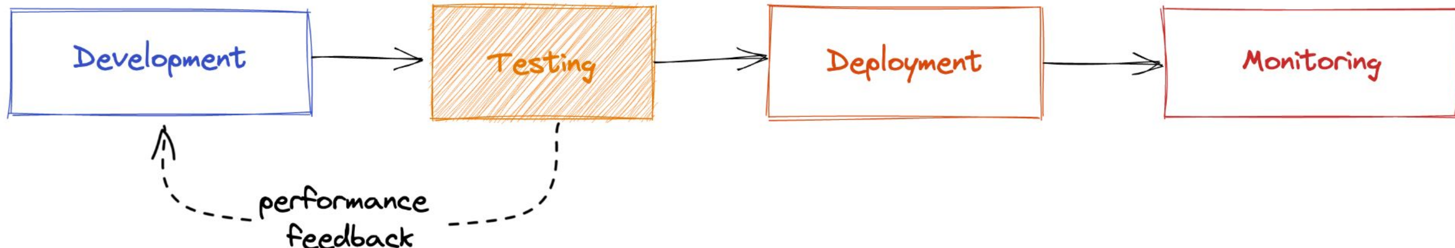
Performance feedback in the SDLC



Problems

- Performance issues identified in production
- Experiments in production environments
- Dev != Ops
- Costs

Shifting left



The ideal workflow

- Performance checks as a testing flavour
- Block merging upon regression
- Reliable performance history
- Compare performance from anywhere

Requirements for the performance metric

- Consistent
- Repeatable
- Hardware agnostic

Measuring performance

The toy algorithm

```
def fibonacci(n: int) -> int:
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)
```

Basic approach

```python
import time

start = time.perf_counter()
fibonacci(10)
end = time.perf_counter()

elapsed_time = round((end - start) * 10**6, 3)
print(f"Elapsed time: {elapsed_time} µs")
```

```
$ python main.py
Elapsed time: 15.84 µs

$ python main.py
Elapsed time: 23.869 µs

$ python main.py
Elapsed time: 18.054 µs

$ python main.py
Elapsed time: 17.041 µs
```

`time.time`: system clock

`time.perf_counter`: high resolution timer

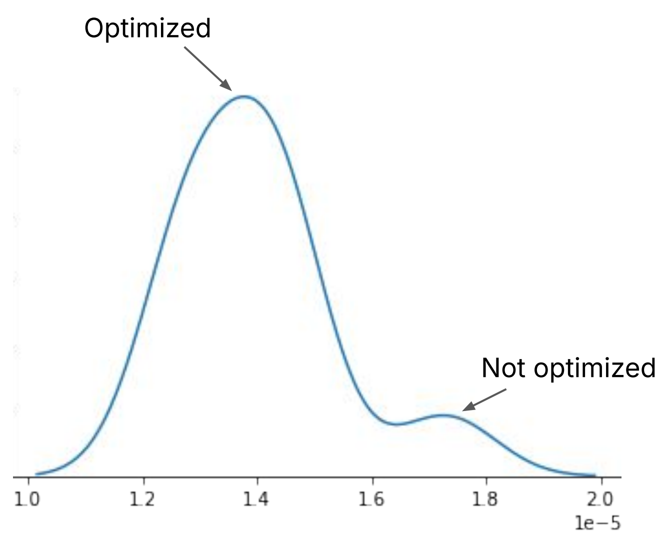
Stats to the rescue

```python
import time

samples = []

for i in range(100):
    start = time.perf_counter()
    fibonacci(10)
    end = time.perf_counter()
    samples.append(end - start)

mean = sum(samples) / len(samples)
mean_micros = round(mean * 10**6, 3)
print(f"Mean: {mean_micros} µs")
```

```
$ python main.py
Mean: 12.46 µs

$ python main.py
Mean: 12.503 µs

$ python main.py
Mean: 13.343 µs

$ python main.py
Mean: 12.218 µs
```

~30% faster

Under the hood



Sample distribution

Solutions:

- Warmup rounds
- Disable garbage collection

Stats to the rescue(with warmup)

```python
import time, gc

gc.disable()  # Disable garbage collection

# Warm up
for i in range(1000):
    fibonacci(10)

# Measure
samples = []
for i in range(100):
    start = time.perf_counter()
    fibonacci(10)
    end = time.perf_counter()
    samples.append(end - start)

mean = sum(samples) / len(samples)
mean_micros = round(mean * 10**6, 3)
print(f"Mean: {mean_micros} µs")
```

```
$ python main.py
Mean: 12.46 µs

$ python main.py
Mean: 12.408 µs

$ python main.py
Mean: 12.243 µs

$ python main.py
Mean: 12.218 µs
```

Improvements: more samples, conditional warmup, outlier removal

Further improvements

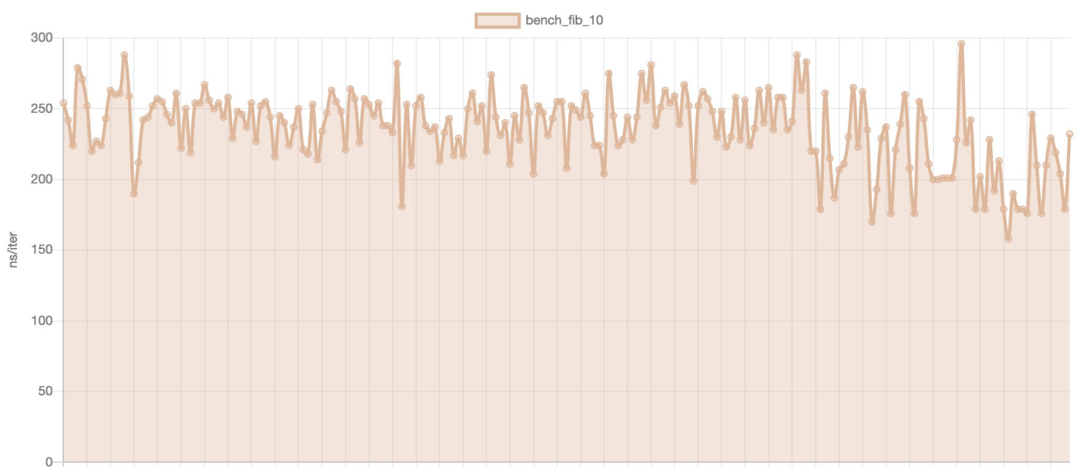
- Conditional warmup
- Remove outliers
- More samples
- Disable the Garbage Collector

Frameworks

- **pytest-benchmark**
- airspeed velocity
- pyperformance

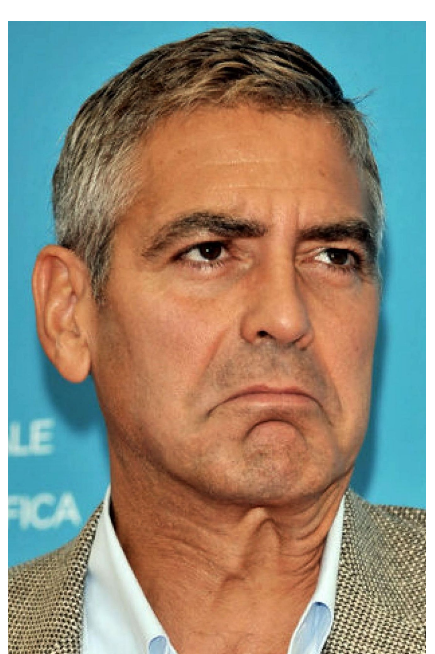
```
def test_fibo_5(benchmark):
    @benchmark
    def _():
        iterative_fibonacci(5)


def test_fibo_10(benchmark):
    @benchmark
    def _():
        iterative_fibonacci(10)


def test_fibo_15(benchmark):
    @benchmark
    def _():
        iterative_fibonacci(15)
```

In a CI environment

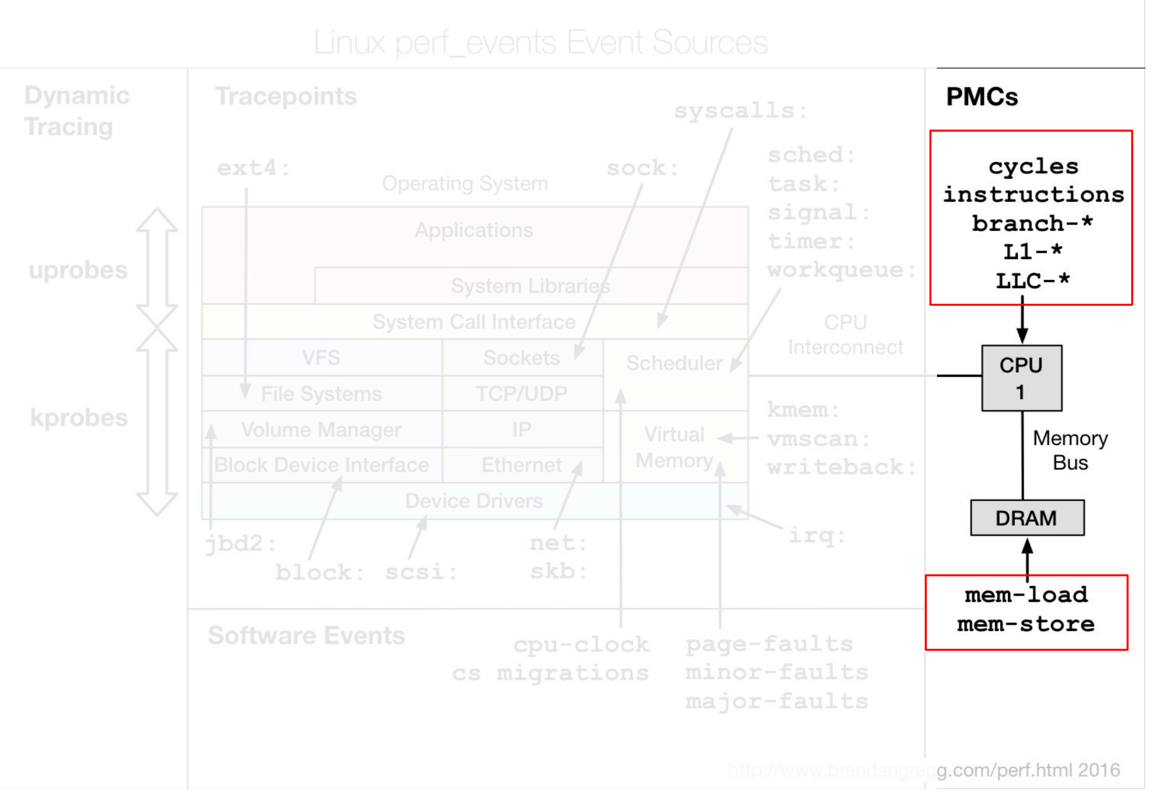


Time measurement for a Fibonacci sequence computation(runs from GitHub Action)

What else?

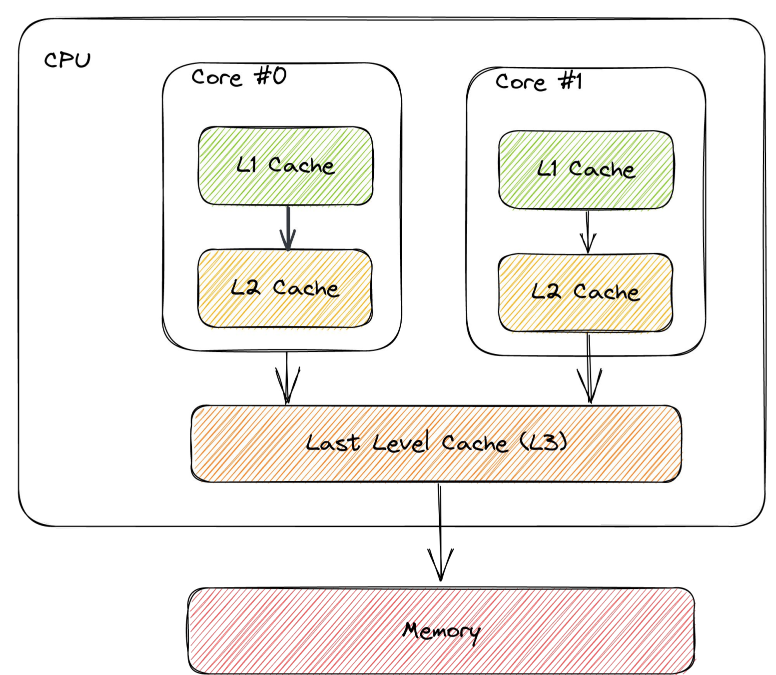


Performance monitoring counters

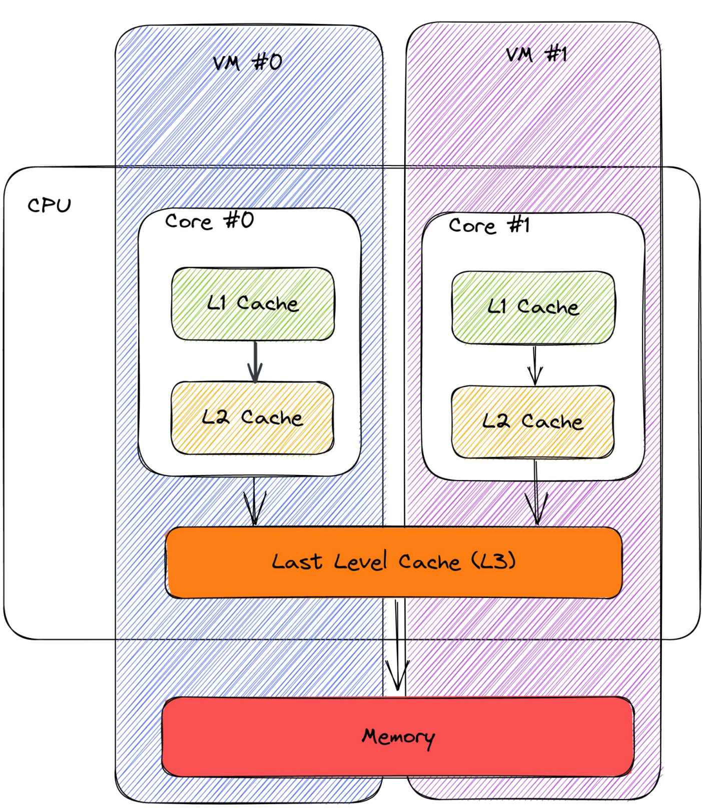


CPU Cache Architecture

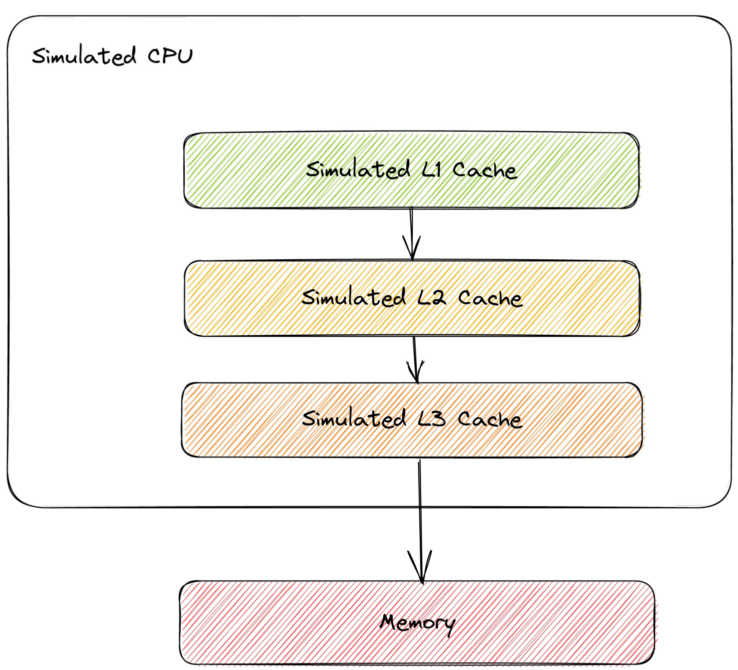
Size	Latency
64KB	1ns
256KB	2.5ns
16MB	11ns
	45ns



Virtualization issue



Simulated caches



Results with CodSpeed

CodSpeed Measurement for a Fibonacci sequence computation(runs from GitHub Action)

Data validation using Python type hints

Strawberry GraphQL

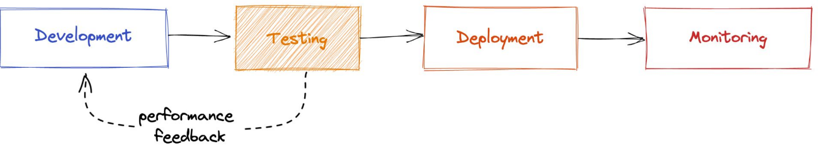
ORMs, ODMs, ...

V2 rewritten in Rust 🦀

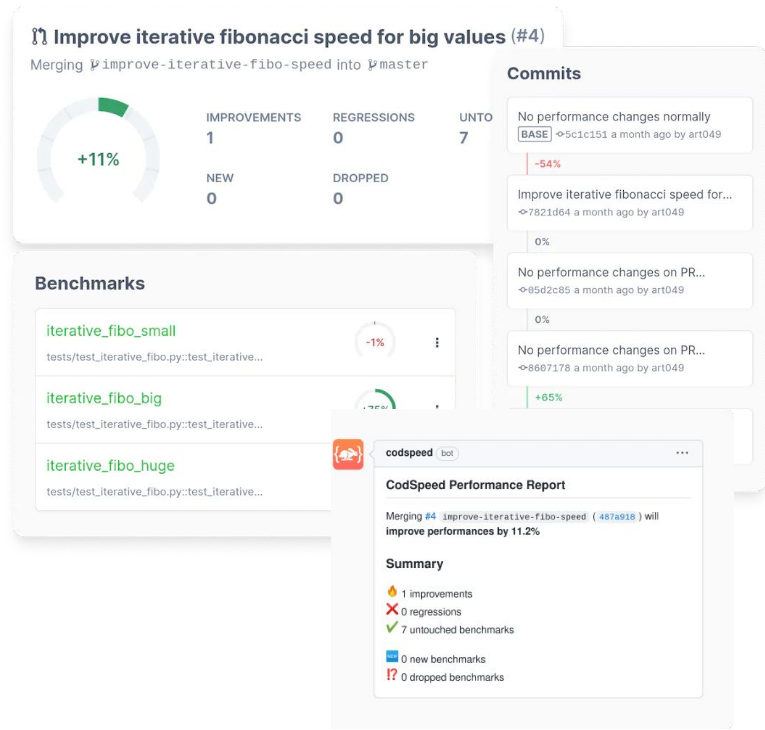
```python
from datetime import datetime
from pydantic import BaseModel


class User(BaseModel):
    id: int
    name = 'John Doe'
    signup_ts: datetime | None = None
    friends: list[int] = []


external_data = {
    'id': '123',
    'signup_ts': '2019-06-01 12:22',
    'friends': [1, 2, '3'],
}
user = User(**external_data)
print(user.id)
#> 123
print(repr(user.signup_ts))
#> datetime.datetime(2019, 6, 1, 12, 22)
print(user.friends)
#> [1, 2, 3]
print(user.dict())
"""
{
    'id': 123,
    'signup_ts': datetime.datetime(2019, 6, 1, 12, 22),
    'friends': [1, 2, 3],
    'name': 'John Doe',
}
"""
```

Demo

Recap

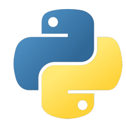


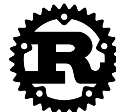
Performance reporting during development



Visibility for future deliveries

Supported languages





Thank you!