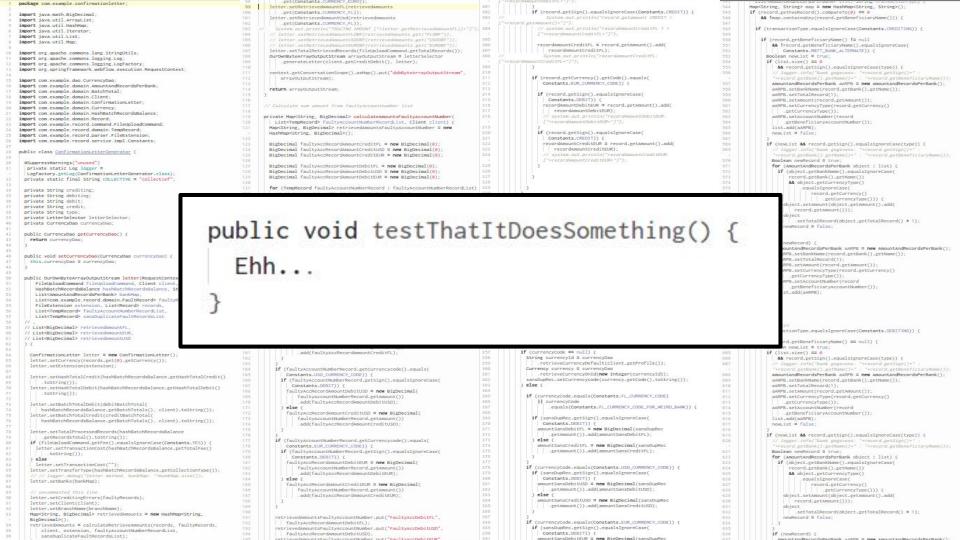
Don't Refactor. Rebuild.

Kinda.

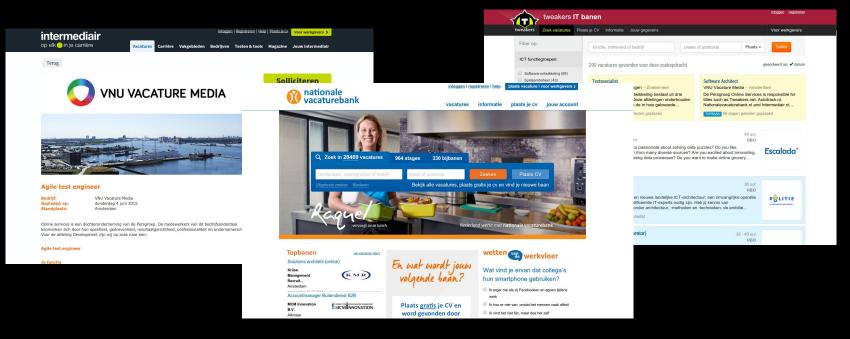
Wouter Lagerweij
@wouterla
Kansas City Developer Conference
#kcdc2017

```
class Dollar {
                                     private int amount;
                                     Dollar(int amount) {
public void testMultiplication() {
                                       this.amount = amount;
    Dollar five = new Dollar(5);
    five.times(2);
    assertEquals(10, five.amount);
                                     void times(int multiplier) {
                                       return amount * multiplier;
```

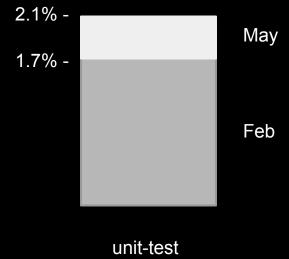




VNU VACATURE MEDIA



- TDD
- BDD
- coding katas
- pairing
- training w/ Chet
- Improve test run time
- 100% coverage of new code
- refactor in the small (boy scout rule)
- refactor in the large (planned improvements)



coverage

25%



Rebuild?



Why you shouldn't

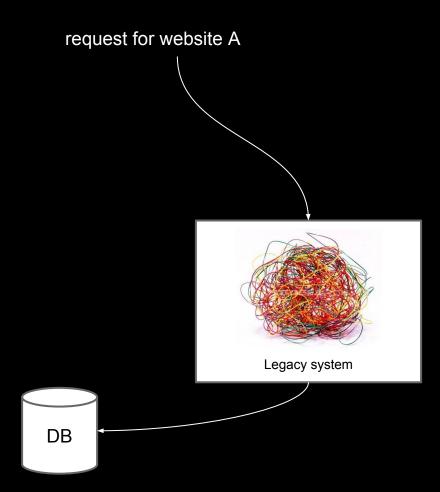
- What was it supposed to do?
- What did it end up doing?
- Do we still want that?
- Why would it be better now?
- No new new things, or playing catch-up?

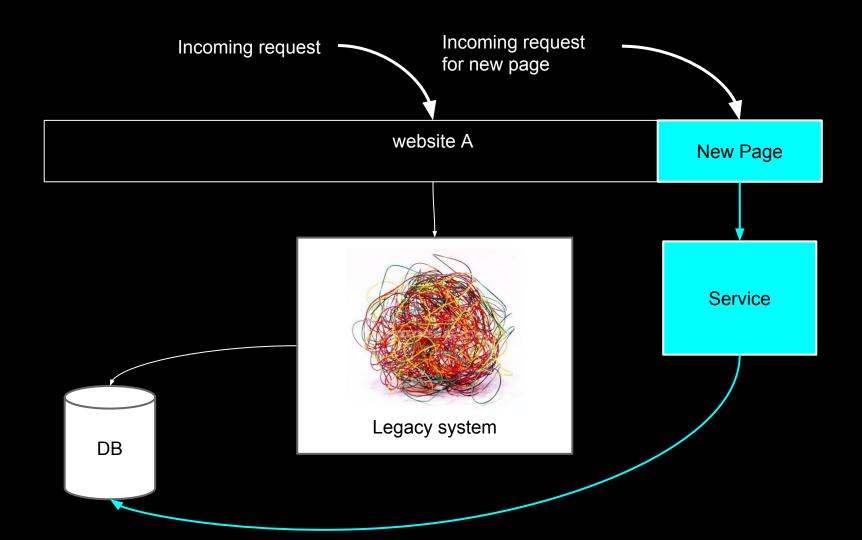
So...

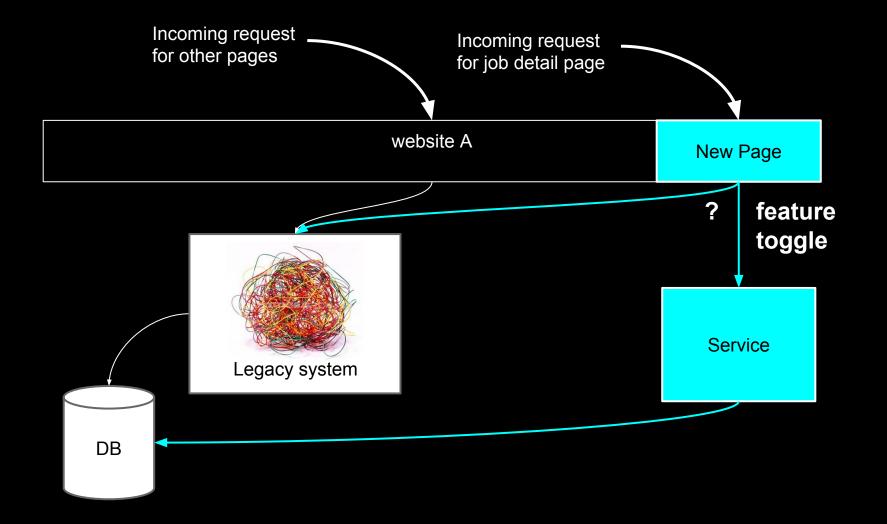
- Deliver new value from day one
- Focus on quality
- Change your process
- Be specific in what you want now

In Practice

- Strangler Pattern
- Continuous Delivery
- BDD









Continuous Delivery

"Every Push To Production"

BDD

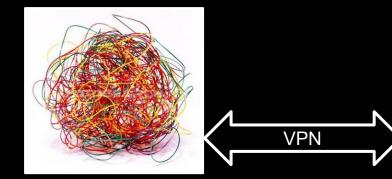


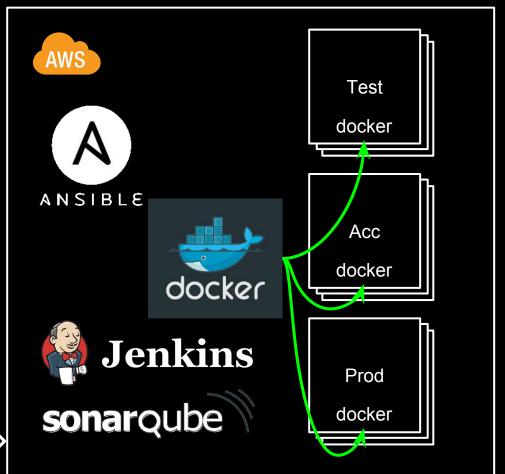






IIII Quay.i₀





In one month

- legacy system
- few tests
- manual deploys
- 1 release per week
- timid team

- decoupled services
- 100% coverage
- full automation
- 30 releases per day
- courage

jobservice

#417 triggered by user _____started 2 hours ago





Thank You!

@wouterla
Wouter Lagerweij
http://www.lagerweij.com/