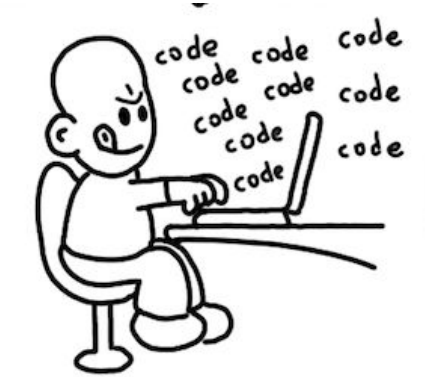
Secrets Behind Writing Specs

Chandra Mohan Thakur
Sr. Software Engineer
CloudFactory

We all know but it's still a secret

People know **what they do**; frequently they know **why they do** what they do; but what they don't know is what **what they do does**."

— Michel Foucault


code
code
code
code
code
code
code
code
code
code


deploy
Done


Bug
Fix
Fix
Fix
Fix
bug
bug
bug
bug
Bug
redeploy

What’s wrong with above scenario?

Why we do this?

- Lack of time and tight schedule
- Fixed budget
- Don’t want to play with legacy code
- Dedicate time to learn testing frameworks
- Spend more time to write spec instead of actual code

What if I don’t write spec?

- Production Code only
- Logics gets concentrated and leads difficult to refactor and test
- Leads to bus factor
- Minor code change leads high risk of failure in dependent features
- Hard to onboard a new developer

After effects of above scenario

- QA needs to perform all test for minor changes
- Buggy production environment
- Project goes out of budget and deadline
- Delay in release cycle

How can we prevent?

Simply write specs for your codes

Why to write specs?

- Provides documentation for your features/library
- Identifies dependent feature bugs easily
- Easy to find unknowns
- Minor changes can be deployed easily(maybe without QA)
- Easy to upgrade technical stuffs
- Less burden while testing features by QA

Continued...

- Quick development cycle time
- QA focus on finding hidden bugs and edge cases instead of generic tests
- Fast release cycle

Wait...

To grab these benefits you need to follow few protocols

Protocols ??

Protocol 1
Write better specs

- Follow best practices
- Write test that catches errors
- Write both happy scenarios and sad scenarios

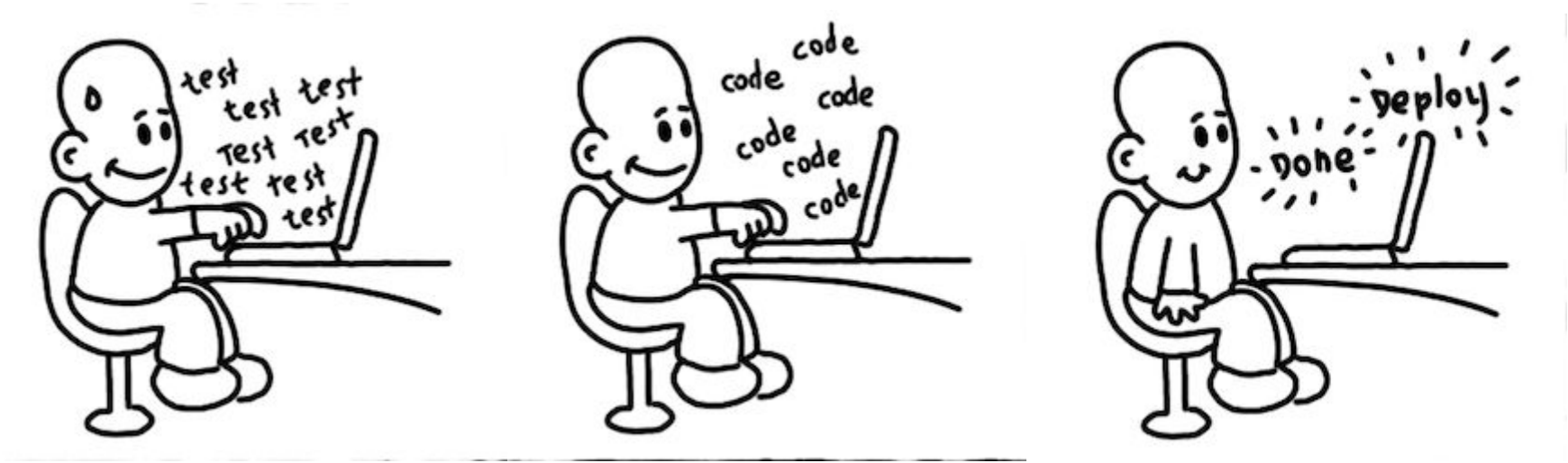
Protocol 2
Follow software development process

TDD

Test-driven development (TDD) is a development technique where you must first write a test that fails before you write new functional code.

BDD

Behavior-driven development (BDD) is a software development methodology in which an application is specified and designed by describing how its behavior should appear to an outside observer.


test
test test
test test
test test
test
code
code
code
code
code
code
deploy
done

So Relax And Let's Catch 'em All

Outcome

- Easy to automate development-testing-deployment
- Easy for Continuous Integration
- Easy for Continuous Delivery
- Documentation for your codes
- Easy to start/deliver new features by developers

Now What?

- If you have new feature
  - Start writing specs
  - Better follow TDD
- If you have enhancement
  - Try to make the code better than before
  - Continuous improvement
- If legacy codebase
  - Add specs for the code you write
  - Follow BDD and write acceptance tests

Thank You

Chandra Mohan Thakur

Twitter: @cmthakur1986
LinkedIn: cmthakur1986