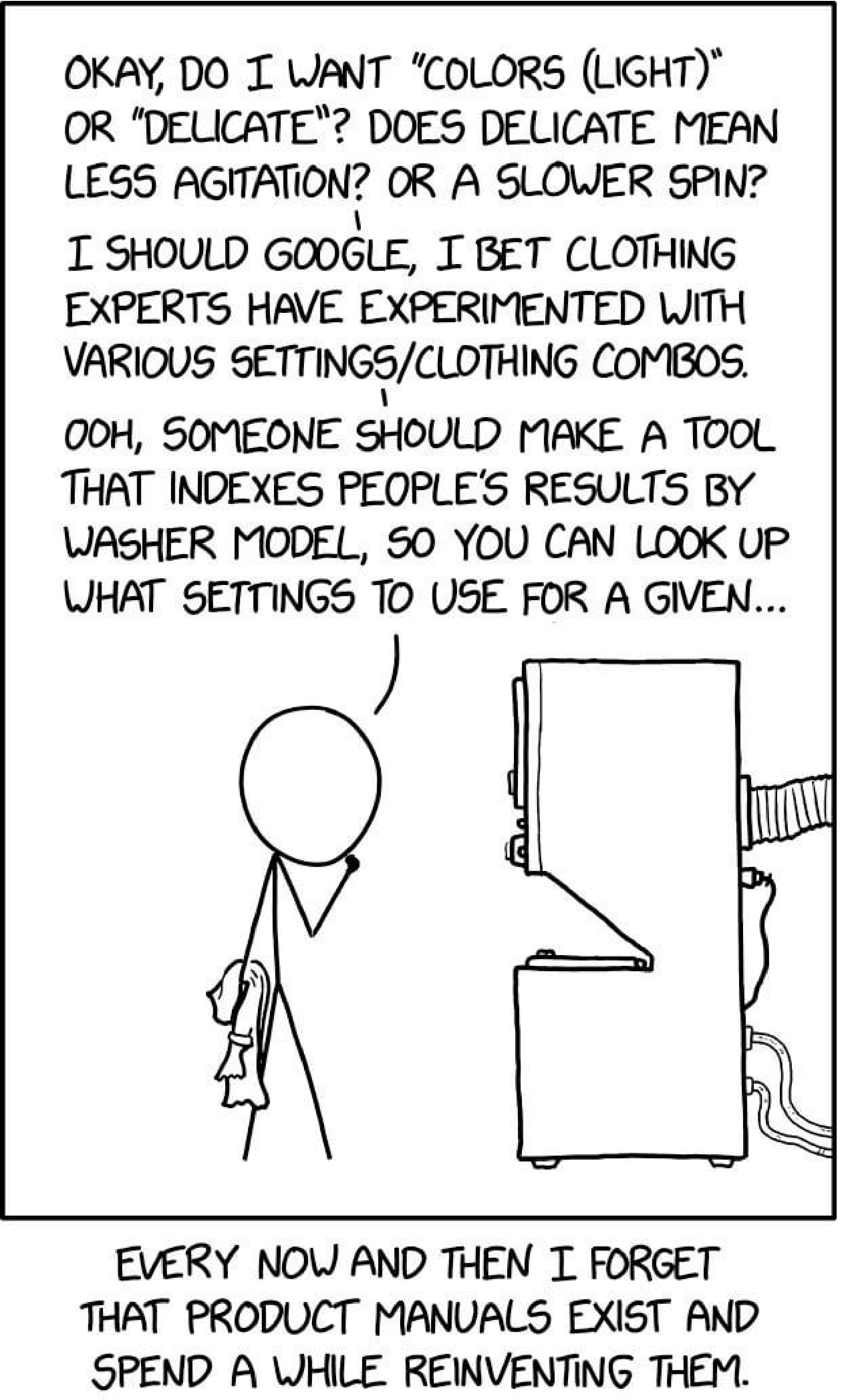

OKAY, DO I WANT "COLORS (LIGHT)" OR "DELICATE"? DOES DELICATE MEAN LESS AGITATION? OR A SLOWER SPIN?
I SHOULD GOOGLE, I BET CLOTHING EXPERTS HAVE EXPERIMENTED WITH VARIOUS SETTINGS/CLOTHING COMBOS.
OOH, SOMEONE SHOULD MAKE A TOOL THAT INDEXES PEOPLE'S RESULTS BY WASHER MODEL, SO YOU CAN LOOK UP WHAT SETTINGS TO USE FOR A GIVEN...
EVERY NOW AND THEN I FORGET THAT PRODUCT MANUALS EXIST AND SPEND A WHILE REINVENTING THEM.

CI/CD and GitHub

WHY DO WE EVEN NEED MARKETING? THEY JUST MAKE EVERYTHING DUMB.
MAN, YOU'RE RIGHT! LET'S START A COMPANY WHERE PRODUCTS ARE DESIGNED, BUILT, AND MARKETED BY ENGINEERS.
Introducing. MAGNATABLET
"If the user interface is confusing, it's because you're stupid."

The Story So Far …

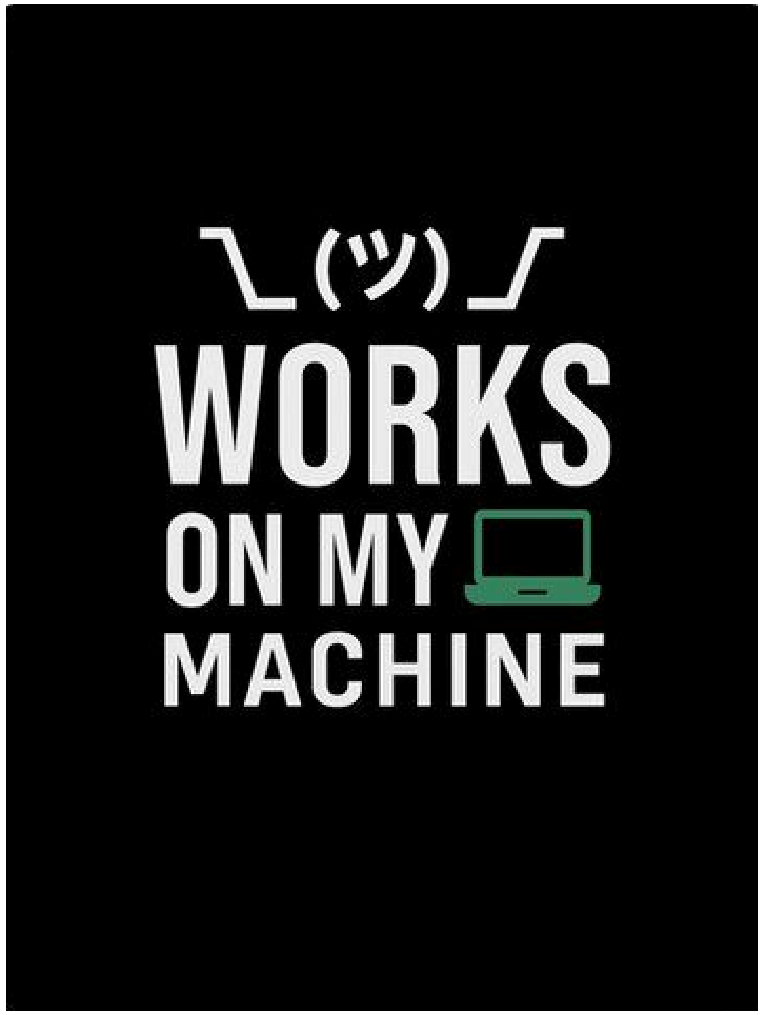
- **Quality assurance** is critical to software engineering.
    - Static and dynamic QA approaches are common
- **A software development process** divides software development into distinct phases to improve design, product, and project management.
- **Process** is the set of activities and associated results that produce a software product.
- How do **QA** and **Process** connect?

One-Slide Summary

- **Continuous Integration / Continuous Deployment (CI/CD)** is a DevOps practice that automates building, testing, and deploying code changes for faster and more reliable software releases.
- **DevOps** is a set of practices, tools, and a cultural philosophy that integrates software development (Dev) and IT operations (Ops) to automate and streamline the software delivery lifecycle.
- The goal is to shorten the time between code changes and production deployment

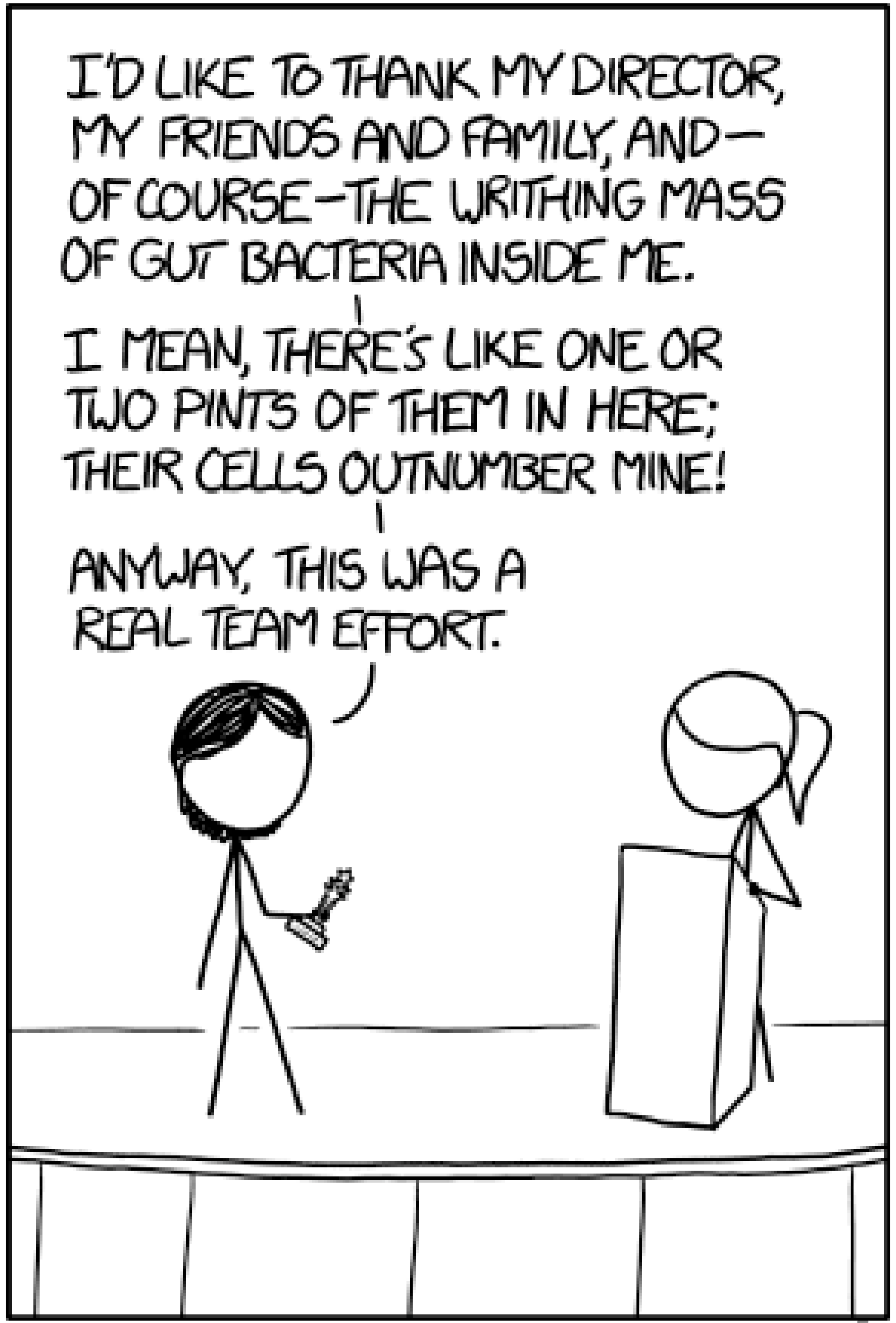
Core Problem

- “Works on my machine”
- Late integration failures
- Manual deployments
- Siloed dev vs ops teams
- Long release cycles



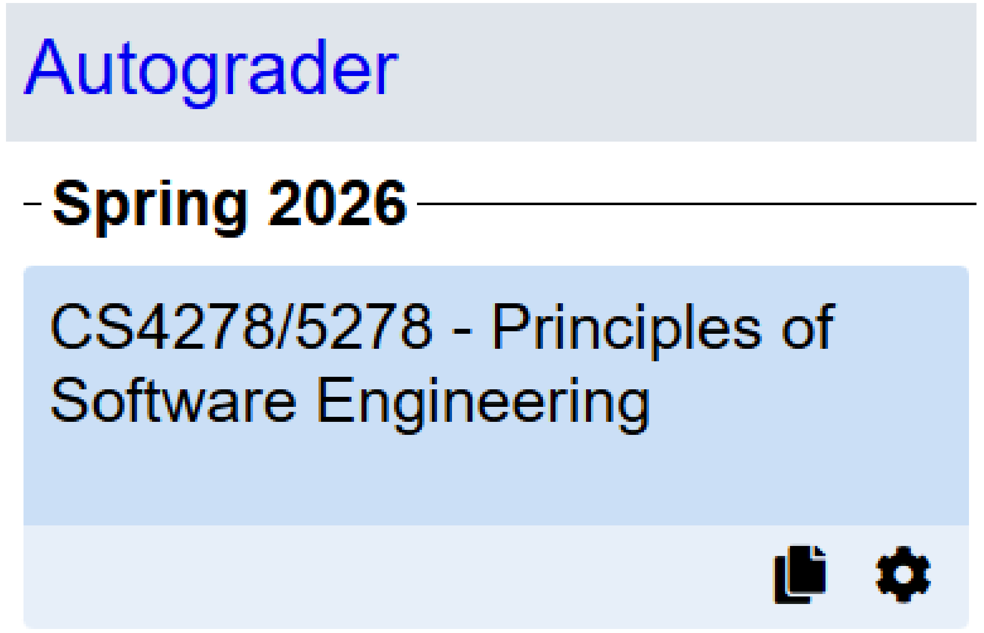
Real Life Motivation

- Group Projects
- “Did my partner do his part yet? I can’t start Part B until he’s at least done with Part A.”
- “Does their part actually work? Did they test that it works with my code?”


I'D LIKE TO THANK MY DIRECTOR, MY FRIENDS AND FAMILY, AND— OF COURSE—THE WRITHING MASS OF GUT BACTERIA INSIDE ME.
I MEAN, THERE'S LIKE ONE OR TWO PINTS OF THEM IN HERE; THEIR CELLS OUTNUMBER MINE!
ANYWAY, THIS WAS A REAL TEAM EFFORT.

Real Life Motivation -- Grading

- Grading takes forever…
- Why not just automatically run all the students' code instead of doing it manually?
- Why not put together a pipeline that runs every time the student submits so they can get instant feedback?


Autograder

Spring 2026

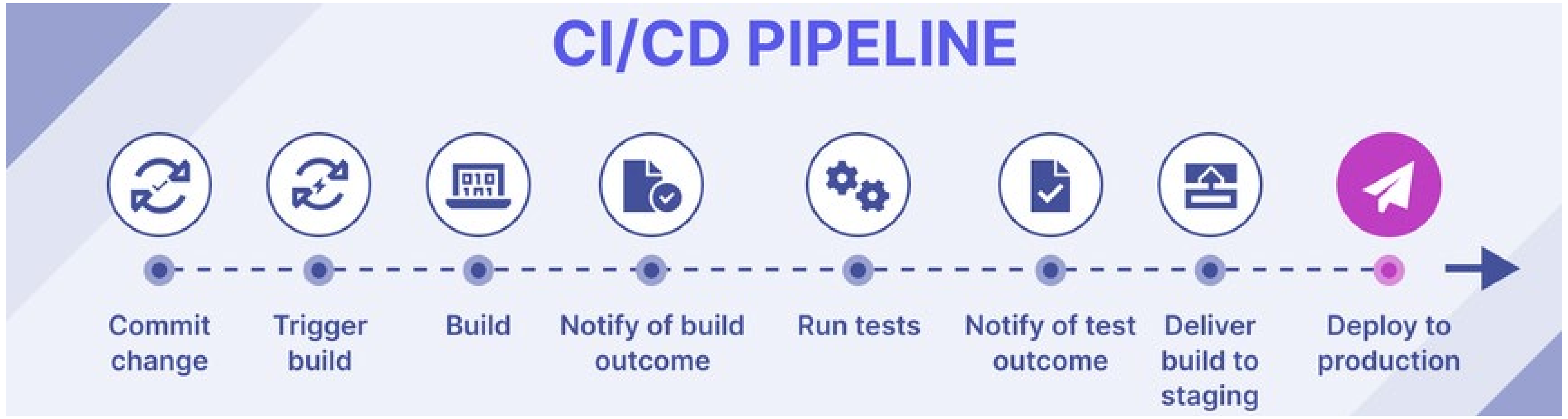
CS4278/5278 - Principles of Software Engineering

Continuous Integration (CI)

- Developers frequently merge code → system automatically builds and runs tests.
- Frequently merge to main
- Automatically build + test on every push
- Fail fast
- Maintain a deployable main branch

Continuous Integration (CI)

- Checkout → Install dependencies → Build → Run tests → Static/Dynamic analysis (linting/coverage) → Artifact creation (reports)



Testing in CI

- Run all the types of tests you are used to (unit, integration, regression)
- Include quality gates so that we don't merge bad code
    - Minimum coverage
    - Linter
    - Static analysis thresholds
    - Security scans
        - Potential secure information leak analysis?
        - Dependency Scanning
        - Supply chain attacks

Common Tools

- Travis CI
- Jenkins
- CircleCI
- **GitHub Actions**
- GitLab CI/CD

Continuous Delivery / Continuous Deployment

- Continuous Delivery = Always Ready to Deploy

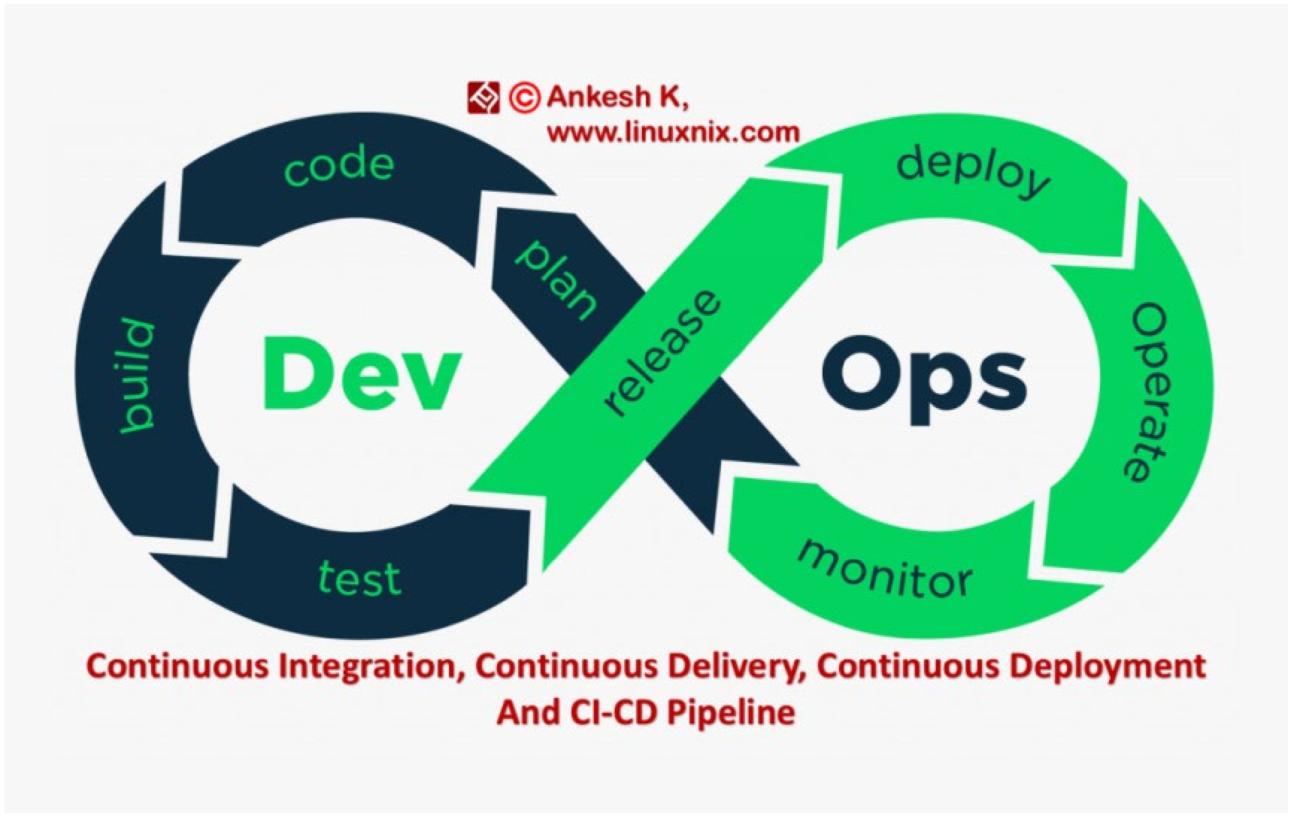
Manual Approval Step

- Continuous Deployment = Automatically deploy to production
    - What could go wrong?

+ Fully Automated
- More Risk

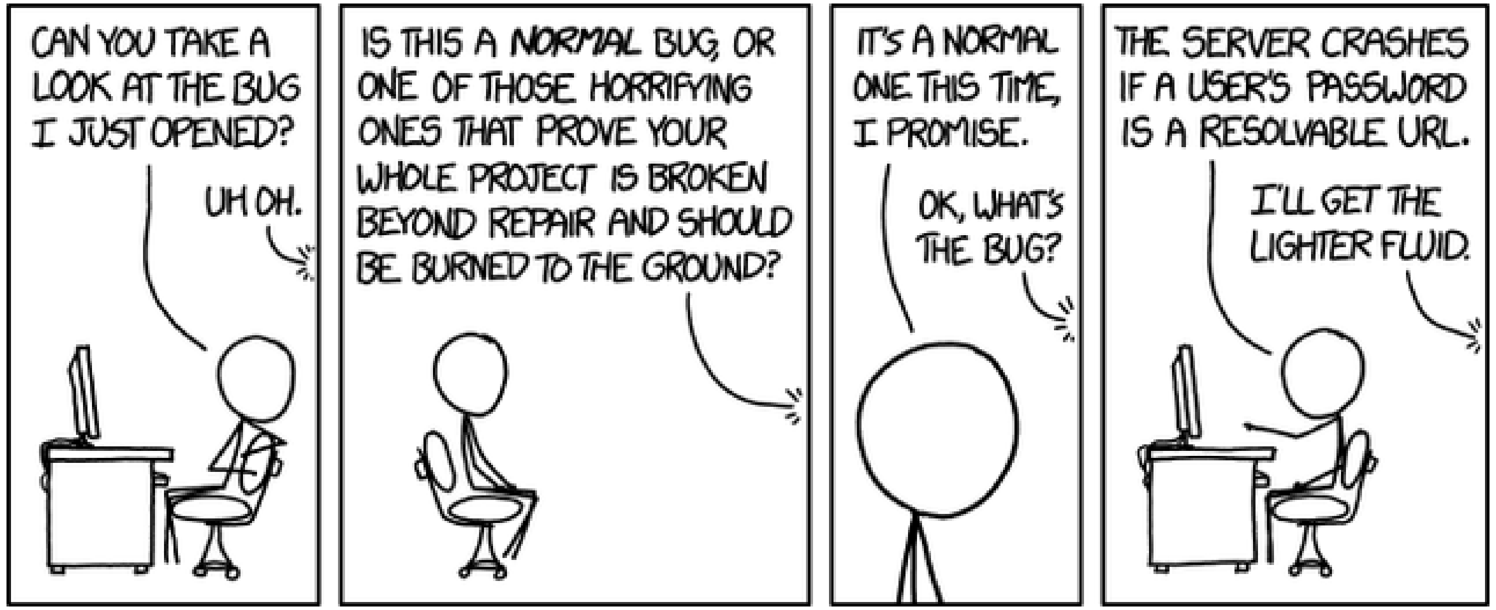
DevOps

- **DevOps** is a set of practices, tools, and a cultural philosophy that integrates software development (Dev) and IT operations (Ops) to automate and streamline the software delivery lifecycle.



DevOps Philosophy

- DevOps is not just tools!
  - Shared ownership
  - Automation-first
  - Monitoring & observability
  - Short feedback loops
  - Infrastructure as Code
  - Blameless postmortems


CAN YOU TAKE A LOOK AT THE BUG I JUST OPENED?
UH OH.
IS THIS A NORMAL BUG, OR ONE OF THOSE HORRIFYING ONES THAT PROVE YOUR WHOLE PROJECT IS BROKEN BEYOND REPAIR AND SHOULD BE BURNED TO THE GROUND?
IT'S A NORMAL ONE THIS TIME, I PROMISE.
OK, WHAT'S THE BUG?
THE SERVER CRASHES IF A USER'S PASSWORD IS A RESOLVABLE URL.
I'LL GET THE LIGHTER FLUID.

DevOps Philosophy

- DevOps is not just tools!
  - Shared ownership
  - **Automation-first**
  - **Monitoring & observability**
  - **Short feedback loops**
  - Infrastructure as Code
  - **“Blameless” postmortems**

These are good! Right…?
cf. pitfalls

Infrastructure as Code

- DevOps often encourages **Infrastructure as Code (IaC)**, the process of managing and provisioning computer/data center resources through machine-readable definition files, rather than physical hardware configuration.
- IaC makes use of
  - Declarative infrastructure
  - Version-controlled configs
  - Reproducible environments
- These can be deployed automatically via the CI/CD pipeline!

Infrastructure as Code

- Docker Containers are like lightweight VMs for a single process or application.
- If shipping a VM is like giving someone your entire computer
  - Shipping a docker container is like giving someone a self-contained application package
  - IaC is like shipping a recipe to build the entire system automatically.

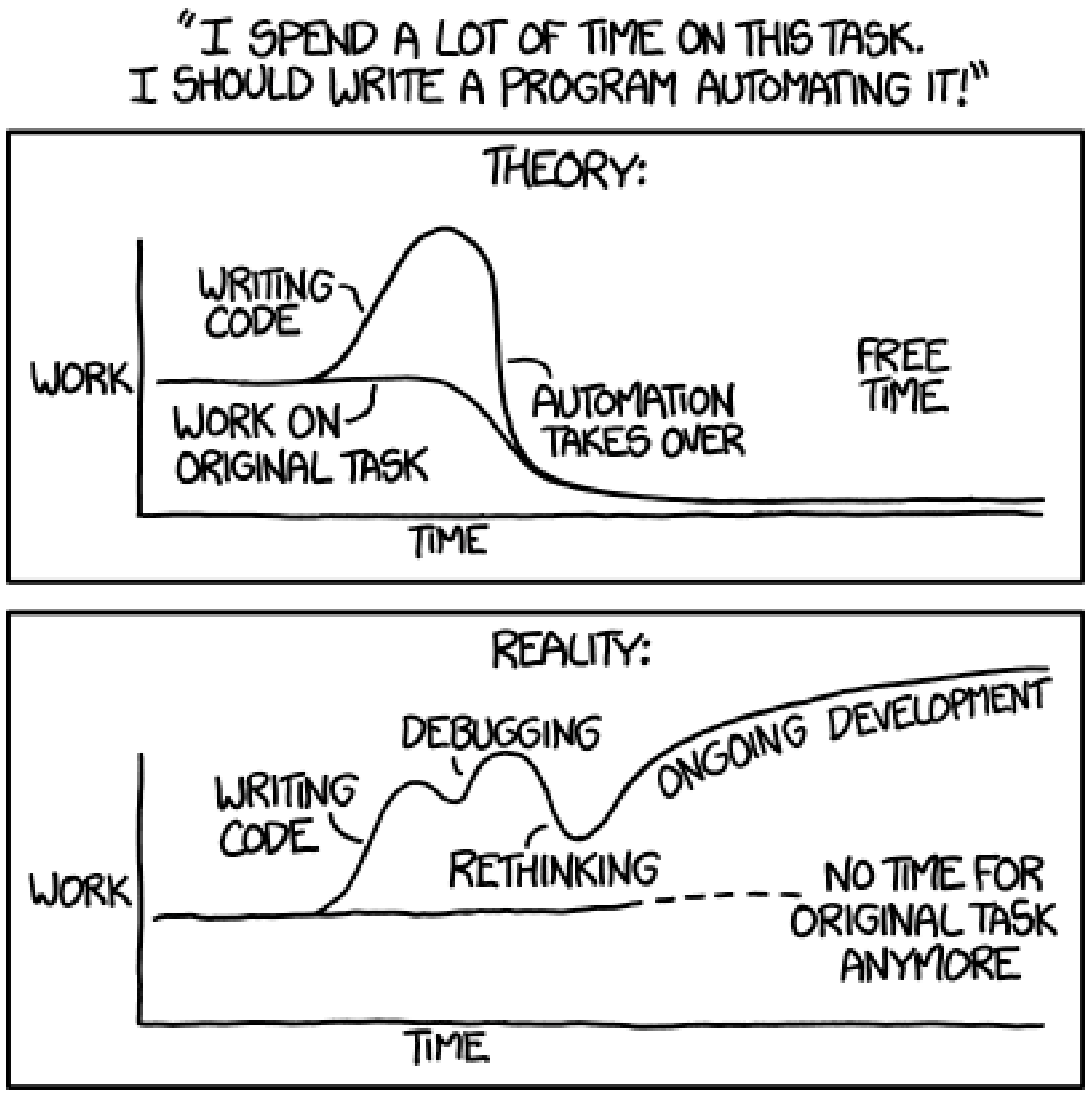
Monitoring and Observability

- CI/CD doesn’t end at deployment
- Monitor
  - Logs| Metrics | Traces | Alerts
  - Crashes | Slowdowns | Traffic
  - MTTR
- Get continuous feedback from production do improve development
  - Project Management and Risk
  - Measurement

DevOps Pitfalls

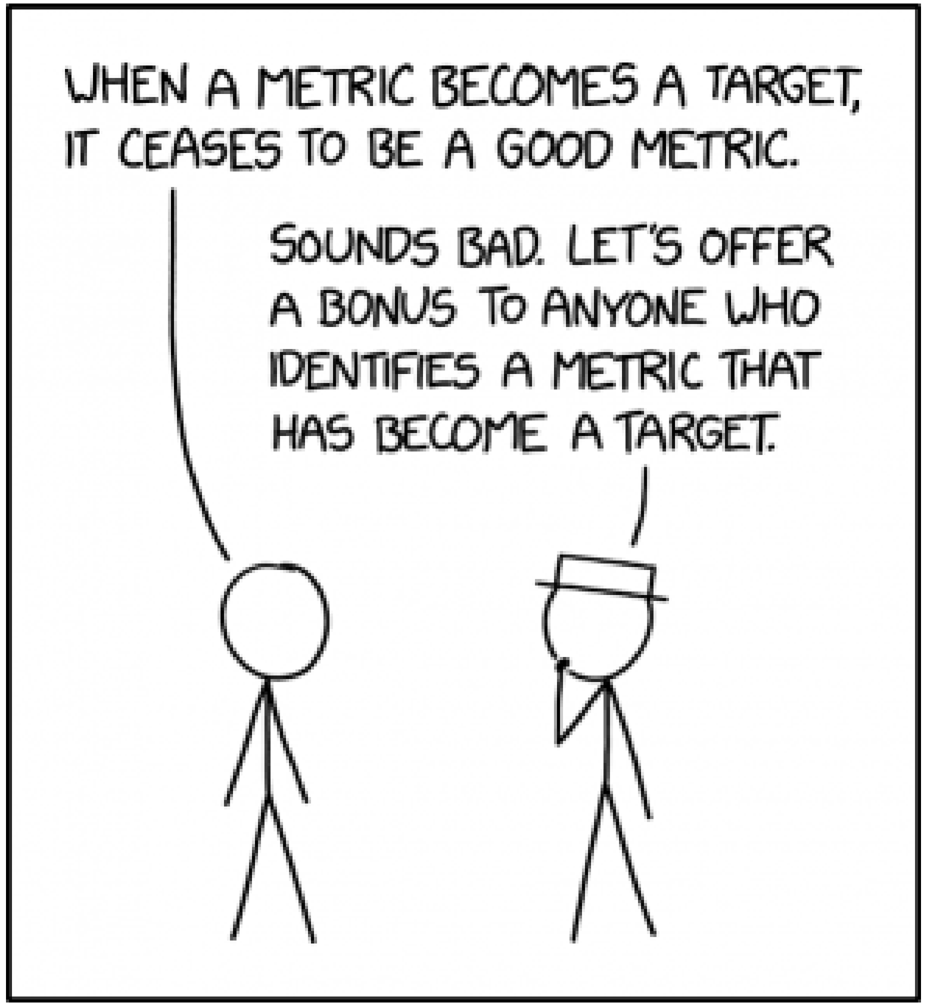
- Automation First

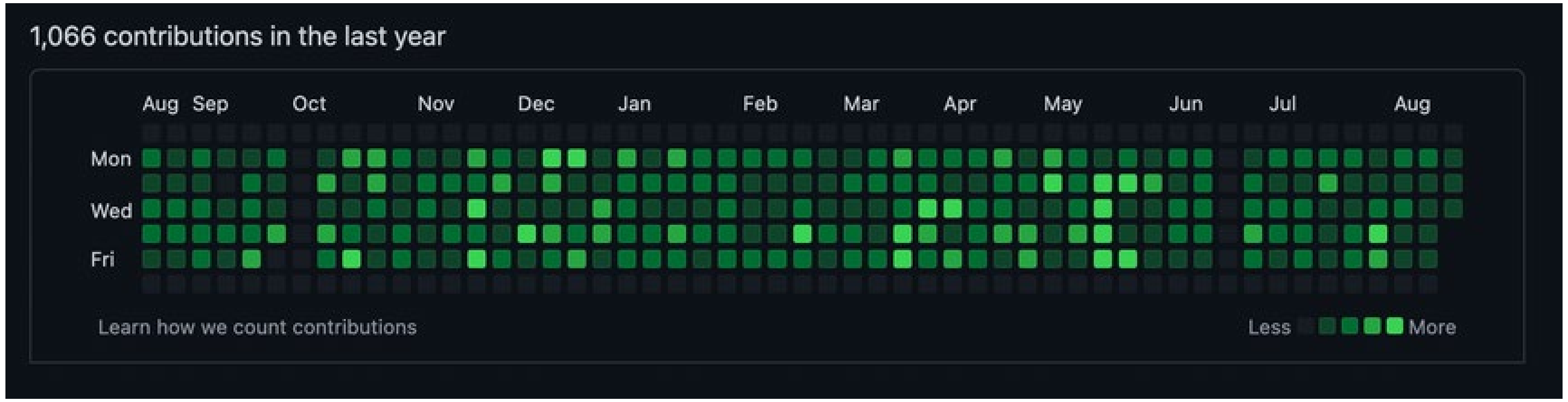
- Don't spend more effort on your automation pipeline than on your program



DevOps Pitfalls

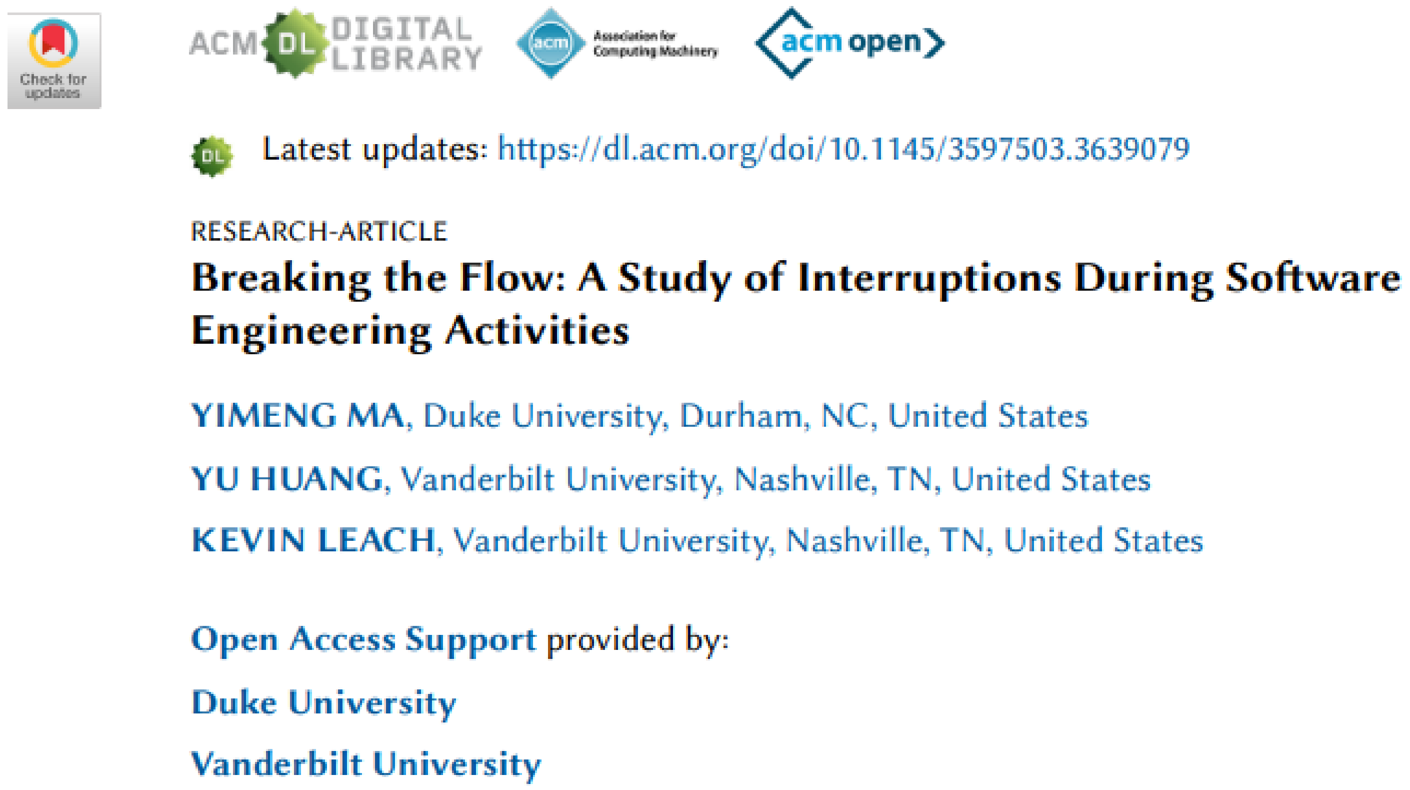
- Monitoring and Observability
- Be careful with metrics!
- Monitoring the entire lifecycle can devolve into monitoring your devs


WHEN A METRIC BECOMES A TARGET, IT CEASES TO BE A GOOD METRIC.
SOUNDS BAD. LET'S OFFER A BONUS TO ANYONE WHO IDENTIFIES A METRIC THAT HAS BECOME A TARGET.


1,066 contributions in the last year
Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug
Mon
Wed
Fri
Learn how we count contributions
Less More

DevOps Pitfalls

- Short feedback cycles
- Do we still have time to get any work done?



Latest updates: https://dl.acm.org/doi/10.1145/3597503.3639079

RESEARCH-ARTICLE

Breaking the Flow: A Study of Interruptions During Software Engineering Activities

YIMENG MA, Duke University, Durham, NC, United States

YU HUANG, Vanderbilt University, Nashville, TN, United States

KEVIN LEACH, Vanderbilt University, Nashville, TN, United States

Open Access Support provided by:

Duke University

Vanderbilt University

DevOps Pitfalls

- “Blameless” Postmortems
- Lol
- Lmao even

Trivia: Toys and Collectibles

- This card game started in 1996 as a simplified spin off of Magic: The Gathering that featured characters from a popular cartoon and Japanese manga.
- With nearly 30 billion cards sold as of 2017, it holds 82% of the card game market share in Europe.
- Gotta catch ‘em all

STAGE 2
Evolves from Charmeleon
Put Charizard on the Stage 1 card
Charizard
120 HP
Flame Pokémon. Length: 5' 7", Weight: 200 lbs.
Pokémon Power: Energy Burn As often as you like during your turn (before your attack), you may turn all Energy attached to Charizard into Energy for the rest of the turn. This power can't be used if Charizard is Asleep, Confused, or Paralyzed.
Fire Spin Discard 2 Energy cards attached to Charizard in order to use this attack.
100
weakness
resistance
-30
retreat cost
Spits fire that is hot enough to melt boulders. Known to unintentionally cause forest fires. LV. 76 #6
Illus. Mitsuhiro Arita
4/102

Open Source Contribution (HW6 for Undergrad)

- Goals:
    - Engage with software engineering
    - Make a meaningful contribution (Pull Request)
    - Reflect on the process and results (Project Report)
- Logistics Overview:
    - 2-student teams are allowed (w/o mixing undergrad and grad)
        - Higher expectations for contribution and project report
    - HW6(A): Task Selection Report (due on 04/02/23)
        - See https://huang.isis.vanderbilt.edu/cs4278/oss4sg.html for ideas
        - Feel free to look around and find a project that you resonate with
    - HW6(B): Project Report (due on 04/18/23)
        - See examples on course website
        - +6% bonus points on HW6(B) if your pull request(s) is/are accepted!

How to make an open source contribution?

- Where to find open source projects?
  - GitHub!!!
- How to spot a good project?
  - Hang on… (next couple of slides)
- How to actually contribute?
  - Pull Requests! (more on this later)

Task Selection

- Find an ***active*** project that is meaningful to you!!
- Where to start?
    - GitHub trending repositories
        - https://github.com/trending?since=monthly
        - Lots of OpenAI-related or ChatGPT-related repositories right now!!
        - Generally very active and fast paced
    - Third Party Monthly Picks
        - https://star-history.com/blog/star-history-monthly-pick-202302
    - Popular Projects (generally very well maintained)
        - Raspberry Pi Projects (https://github.com/raspberrypi)
        - Hyperledger Foundation Projects (https://github.com/orgs/hyperledger)
        - Kubernetes Projects (https://github.com/kubernetes)
        - Google Project, Microsoft Project, etc.

Task Selection Cont.

- Python Projects:
  - TensorFlow
  - OpenCV
  - Flask
- C++ Projects:
  - Microsoft Cognitive Toolkit
  - IncludeOS
  - Kodi
- Java Projects
  - Jenkins
  - Elasticsearch
- Lots of online articles/blogs that can guide you to finding a good project!
  - e.x. https://www.rocket.chat/blog/open-source-projects

Task Selection Advice

- Choose an ***active*** project with many contributors!
- Scope the project well (don’t get overly ambitious)
- Choose one large task or several smaller tasks
- Read the entire homework description on the course website!!!

Task Selection Advice

- Once you identify a task to do, claim it!
  - Especially important for well-maintained projects
  - Someone may already be on the task!

- Create a timeline (both for yourself and for the report)
  - Try to stick to it!

- Start early!!
  - Especially if you want your pull request to get accepted before the deadline.
  - (only PRs accepted before the deadline will get extra-credit)

Project Report Overview

- NO LATE SUBMISSION for HW6(B)
- NO excuse will be accepted!

Project Report Overview

- Show us what you did!
    - Be proud of your contribution!!
- Explain your strategy/approach
- Share your engineering experience
    - What issues/roadblocks did you encounter?
    - How was communication with other community members?
    - How did you fix the bug or make an improvement?
    - Show some evidence of your work :)
- Compare your initial plan to what you have achieved
    - Any differences?
- Many examples on course website!!

Final Remarks

We hope you can have some fun with open source contribution, as it is a vital component of the software engineering community. Maybe you’ll become a regular open source contributor in the future!

Research Proposal (HW6 for Grad)

- Format based on the NSF requirements
- 5-7 pages excluding references
- Use the provided LaTeX template

Research Proposal (HW6 for Grad)

- First Page: Project, Summary, Intellectual Merit, Impact
- Introduction
- Background and Related Work
- Proposed Research
- Proposed Experiments
- Preliminary Work
- Conclusion
- References (IEEE format)

How to make an open source contribution?

- How to actually contribute?
  - Pull Requests! (lots of tutorials online)
    - Fork the repository
    - Clone the repository to local machine (git clone)
    - Create a new branch (git checkout -b [branch-name])
    - Make the changes
    - Commit the changes (git commit)
    - Push the changes (git push)
    - Create a pull request on GitHub UI

In-Class Activity

- Partner up

- Partner 1
  - Create a **public** github repo
  - Add a text document to it
  - Add a GitHub actions CI pipeline

- Partner 2
  - Fork the repo
  - Change the text document
  - Make a pull request to merge changes to main
  - GitHub actions should automatically run

- Partner 1 then accepts the pull request to confirm the merge
- Submit a link to your public repo to Brightspace

To make a pull request
- Fork the repository
- Clone the repository to local machine (git clone)
- Create a new branch (git checkout -b [branch-name])
- Make the changes
- Commit the changes (git commit)
- Push the changes (git push)
- Create a pull request on GitHub UI