Lab 4: CI & CD & DevOps

何昊

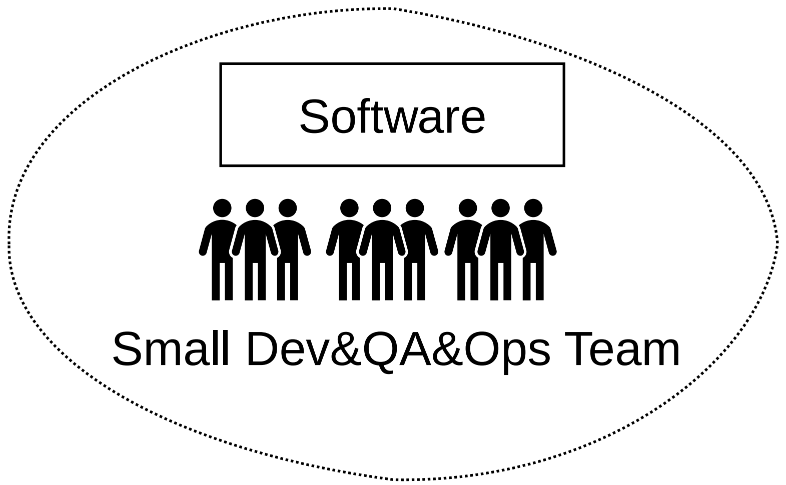
2022.10.15

Some slides are adjusted from:

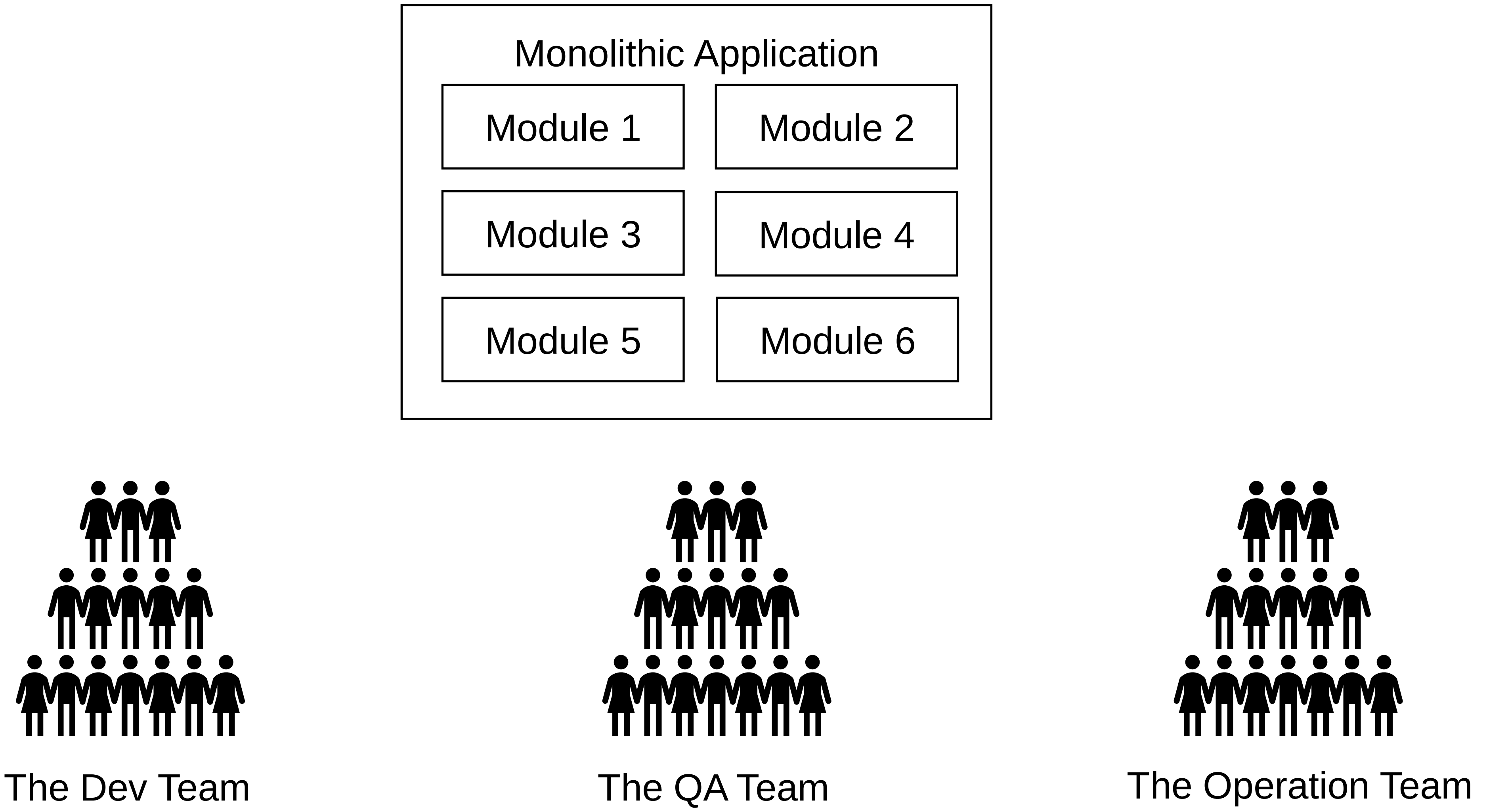
https://cmu-313.github.io/2020/lectures/15-Devops.pdf

The "Software Development" in Your Course Projects

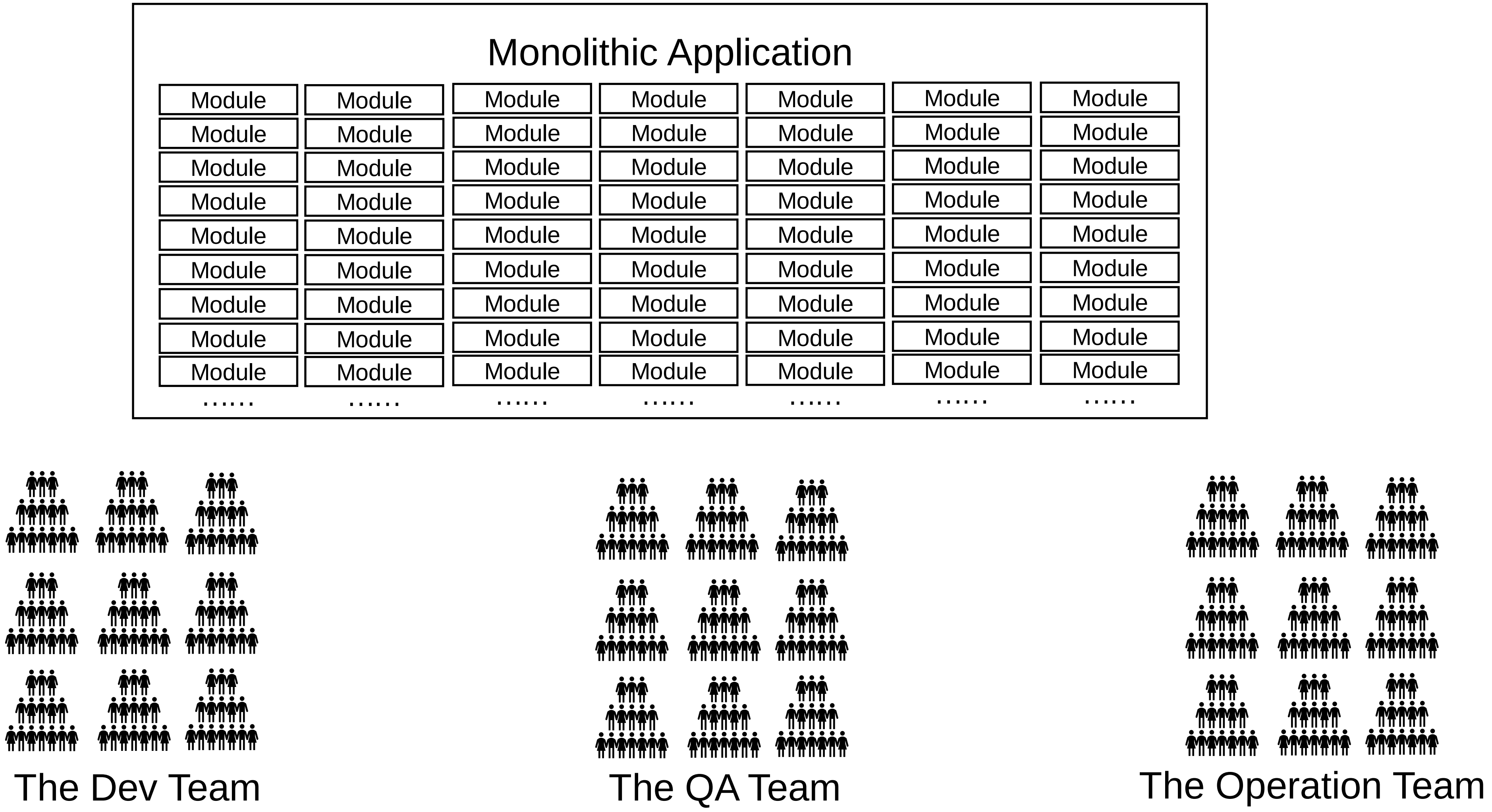
- Small team
- Not too much code
- Everyone takes all roles (development, quality assurance, operation)
- Agile



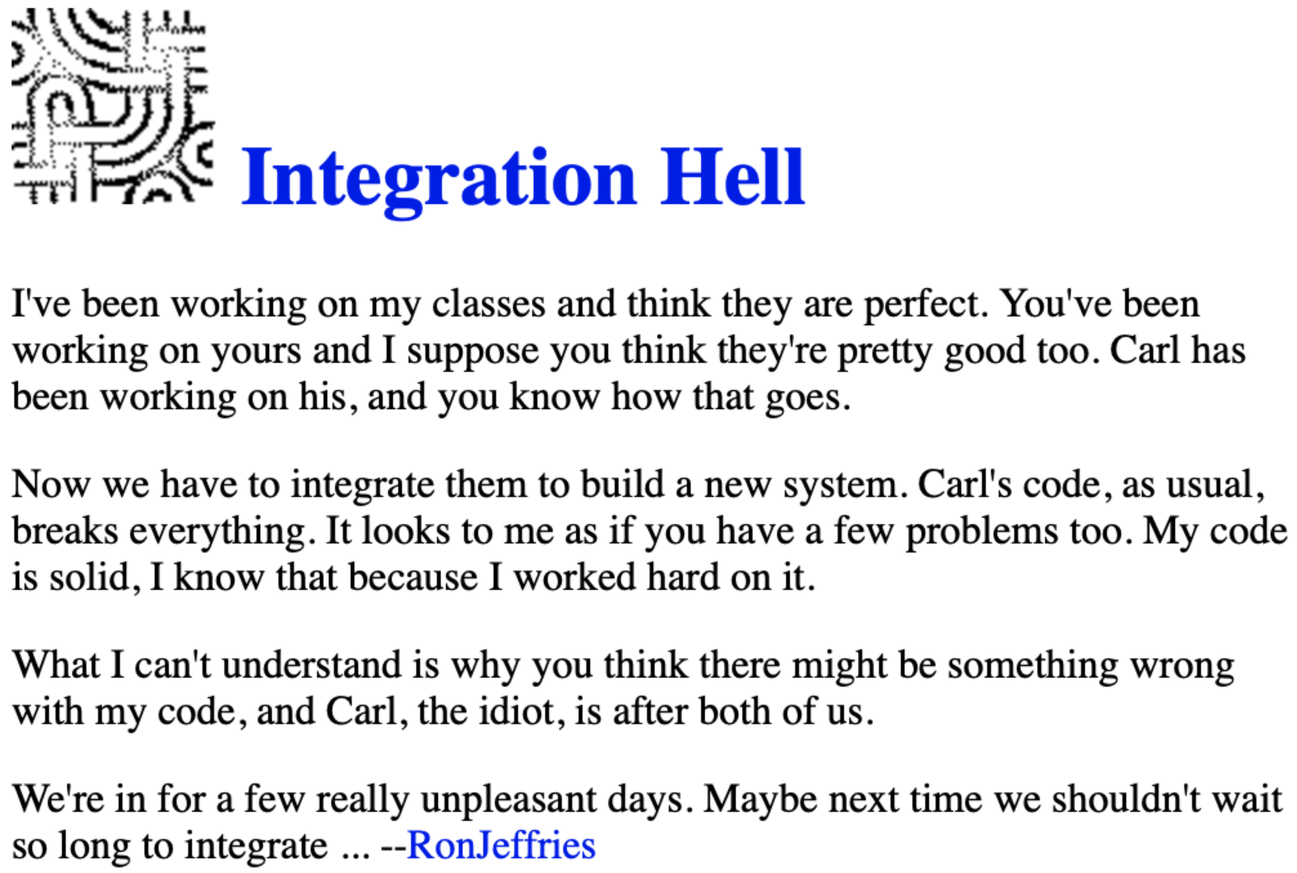
The (Traditional) Software Development



The (Traditional) Software Development


Monolithic Application
Module Module Module Module Module Module Module
Module Module Module Module Module Module Module
Module Module Module Module Module Module Module
Module Module Module Module Module Module Module
Module Module Module Module Module Module Module
Module Module Module Module Module Module Module
Module Module Module Module Module Module Module
Module Module Module Module Module Module Module
Module Module Module Module Module Module Module
....
The Dev Team
The QA Team
The Operation Team

It is All About **Scalability**!



Integration Hell

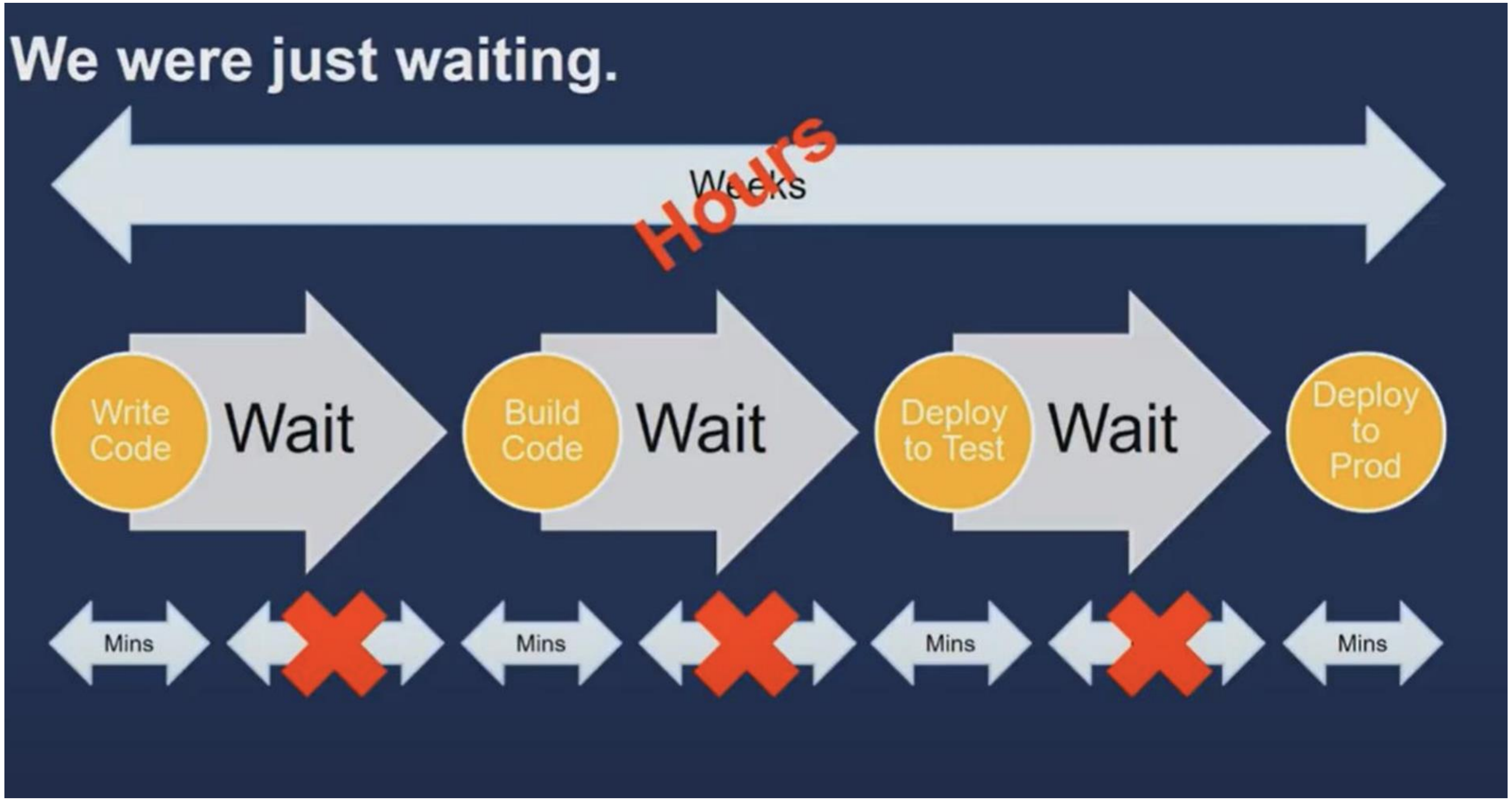
I've been working on my classes and think they are perfect. You've been working on yours and I suppose you think they're pretty good too. Carl has been working on his, and you know how that goes.

Now we have to integrate them to build a new system. Carl's code, as usual, breaks everything. It looks to me as if you have a few problems too. My code is solid, I know that because I worked hard on it.

What I can't understand is why you think there might be something wrong with my code, and Carl, the idiot, is after both of us.

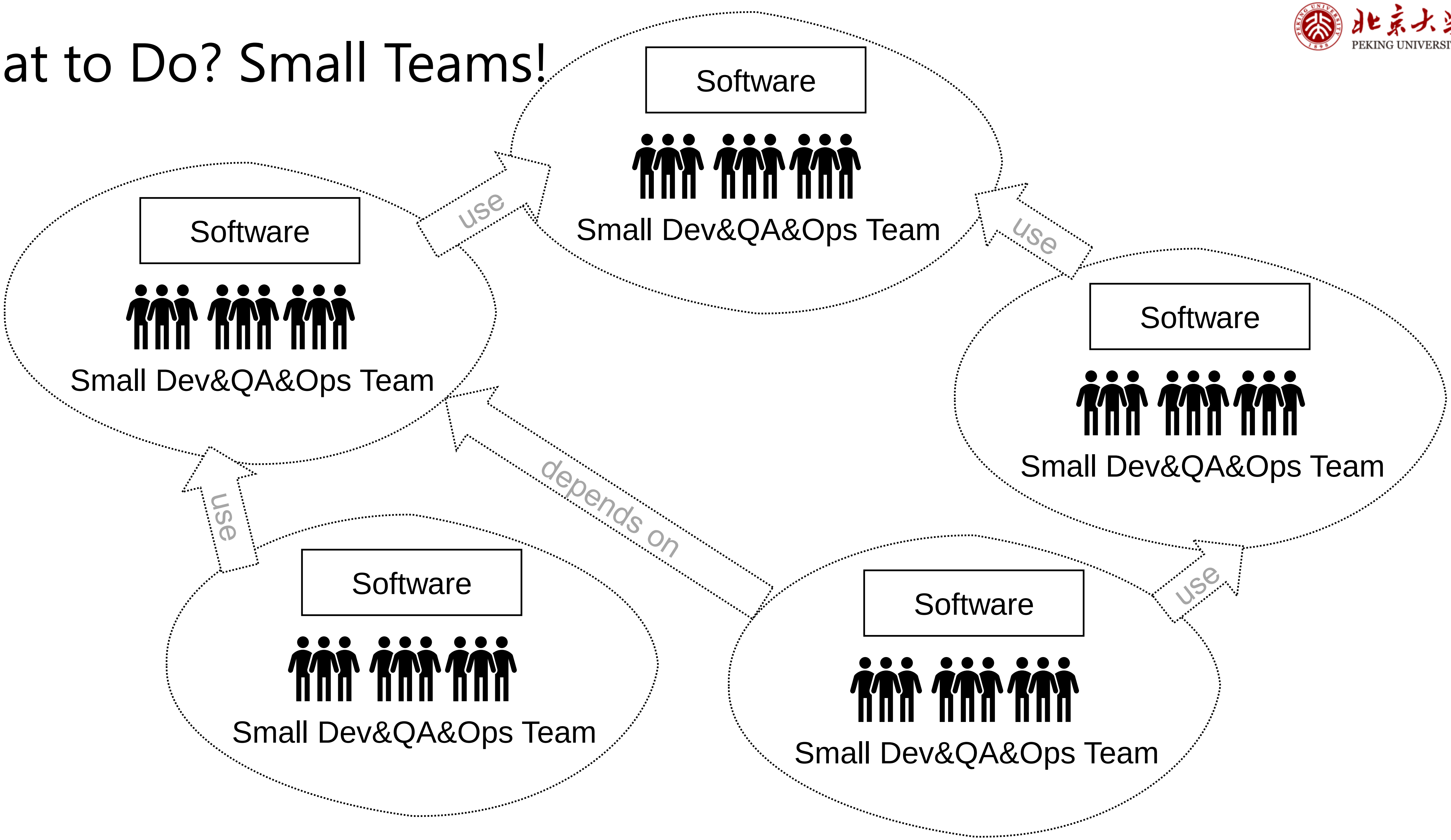
We're in for a few really unpleasant days. Maybe next time we shouldn't wait so long to integrate ... --RonJeffries

It is All About **Scalability**!

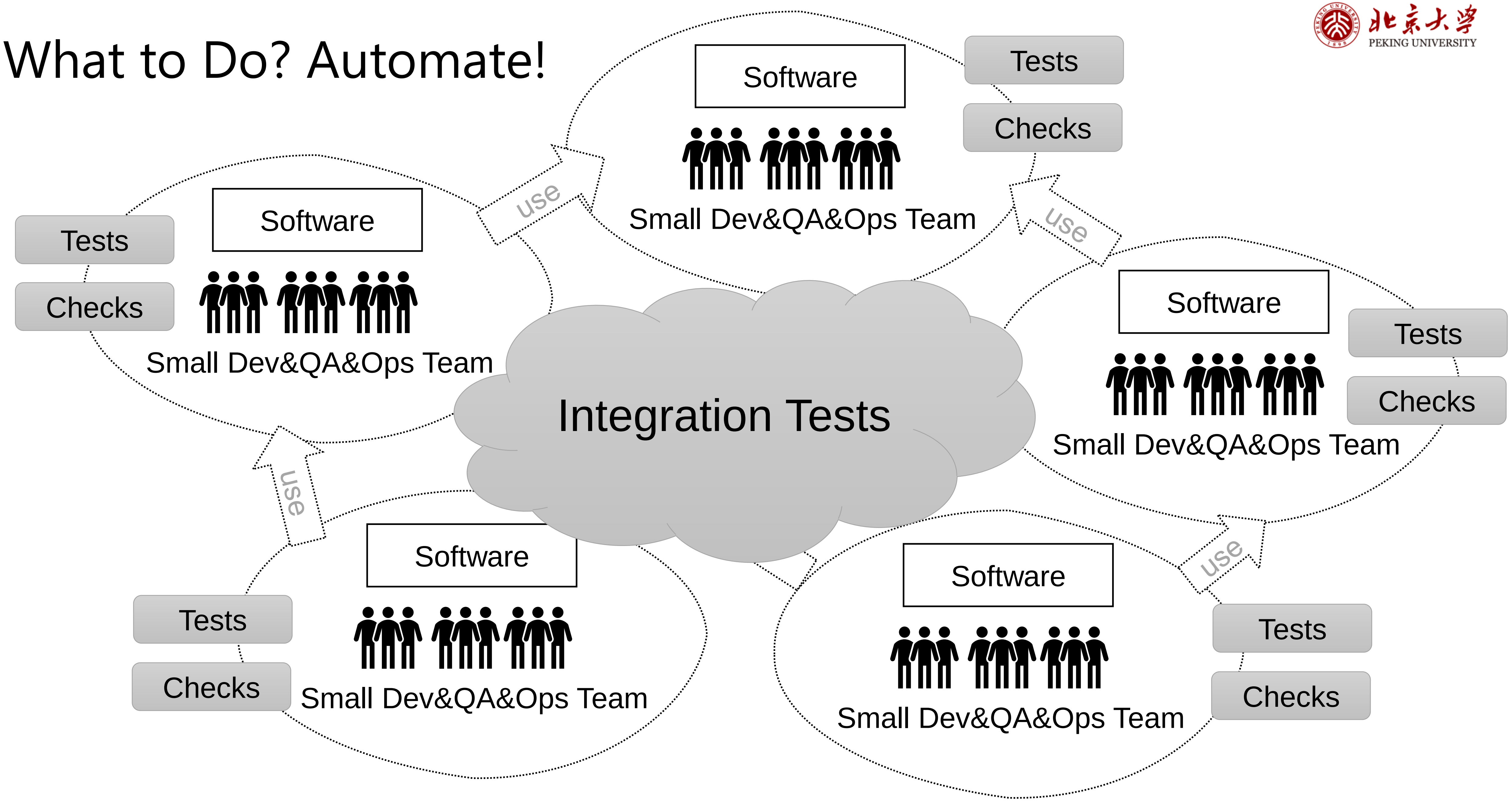


https://www.youtube.com/watch?v=mBU3AJ3j1rg

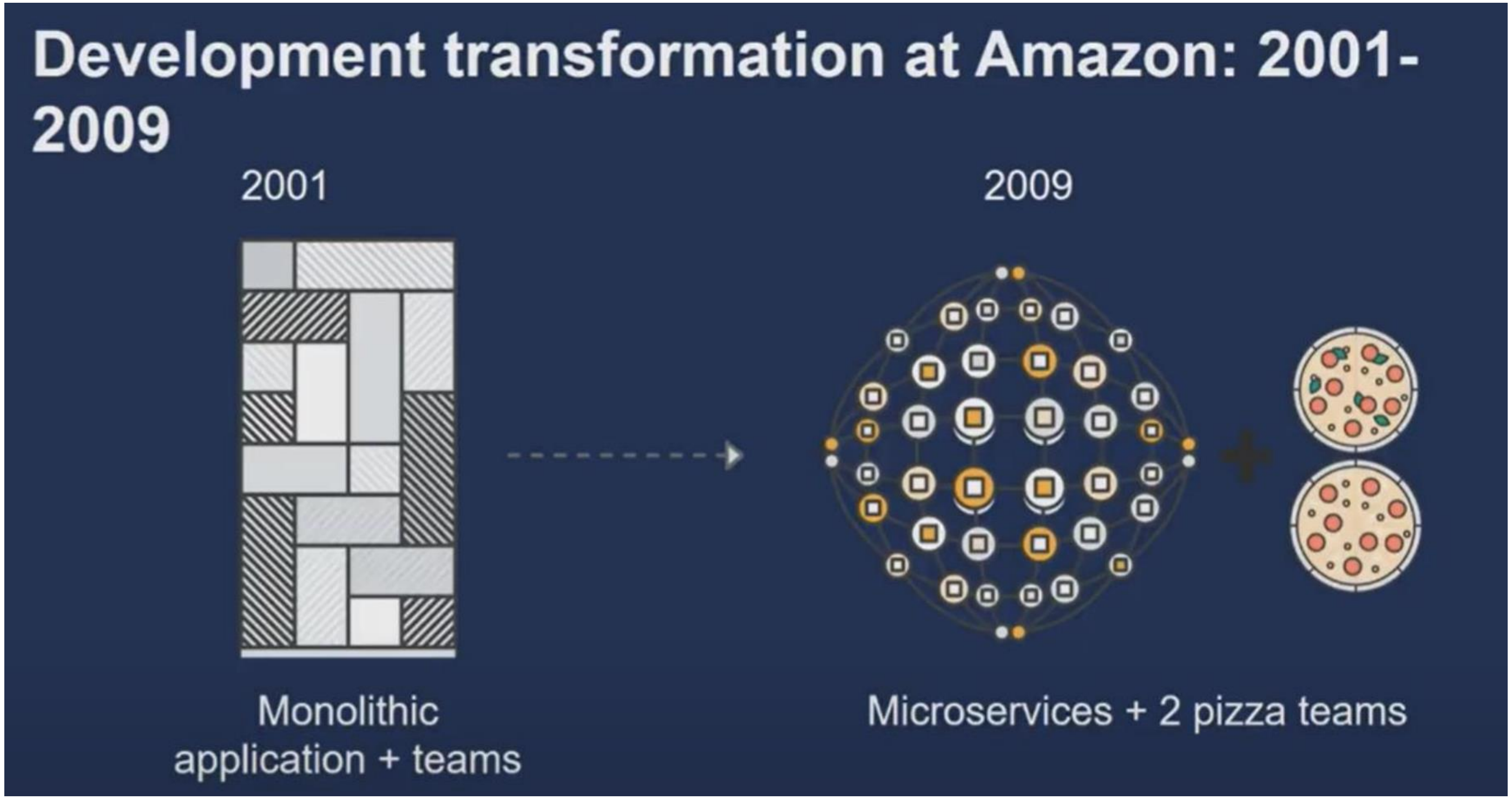
What to Do? Small Teams!



What to Do? Automate!

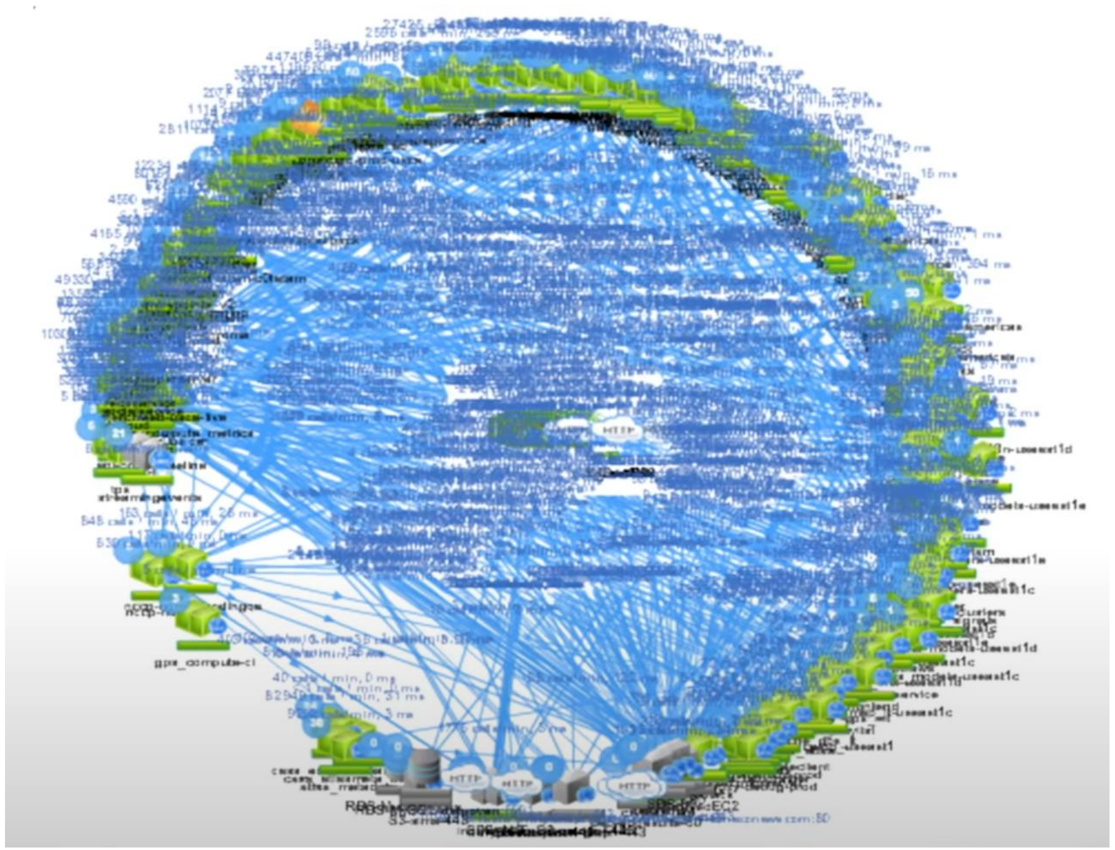


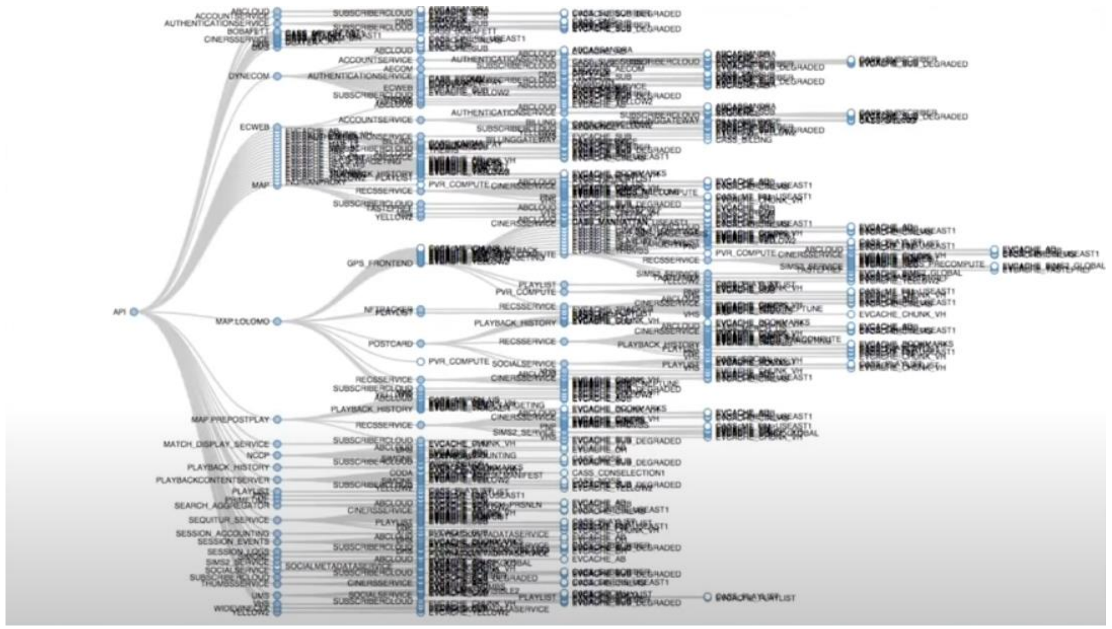
The "Microservice" in Industry


Development transformation at Amazon: 2001-2009
2001
2009
Monolithic application + teams
Microservices + 2 pizza teams

https://www.youtube.com/watch?v=mBU3AJ3j1rg

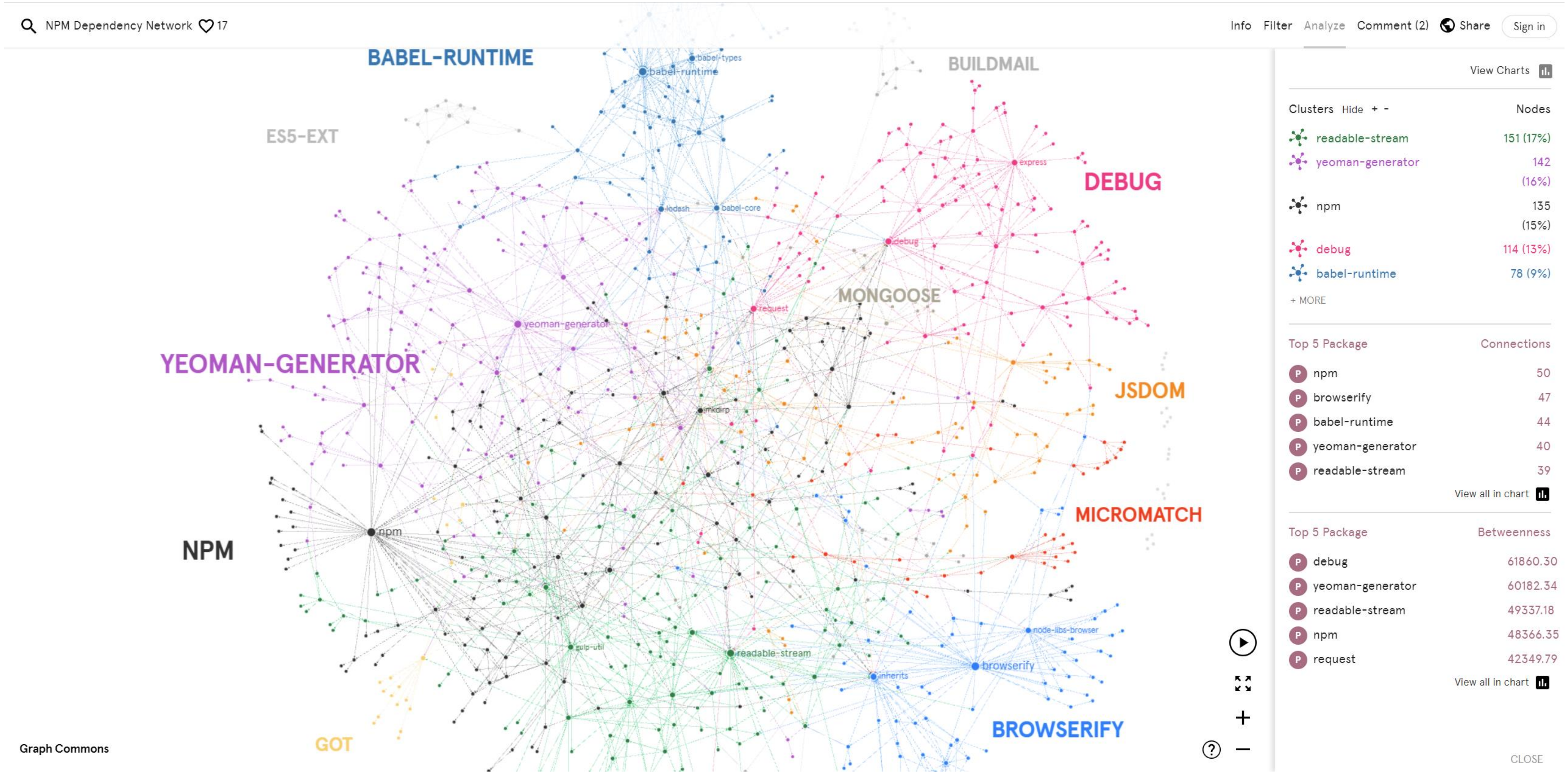
The "Microservice" in Industry





As of Netflix in 2018, https://www.youtube.com/watch?v=UTKIT6STSVM

The "Software Ecosystem" in Open Source



Clusters	Nodes
readable-stream	151 (17%)
yeoman-generator	142 (16%)
npm	135 (15%)
debug	114 (13%)
babel-runtime	78 (9%)

+ MORE

Top 5 Package	Connections
npm	50
browserify	47
babel-runtime	44
yeoman-generator	40
readable-stream	39

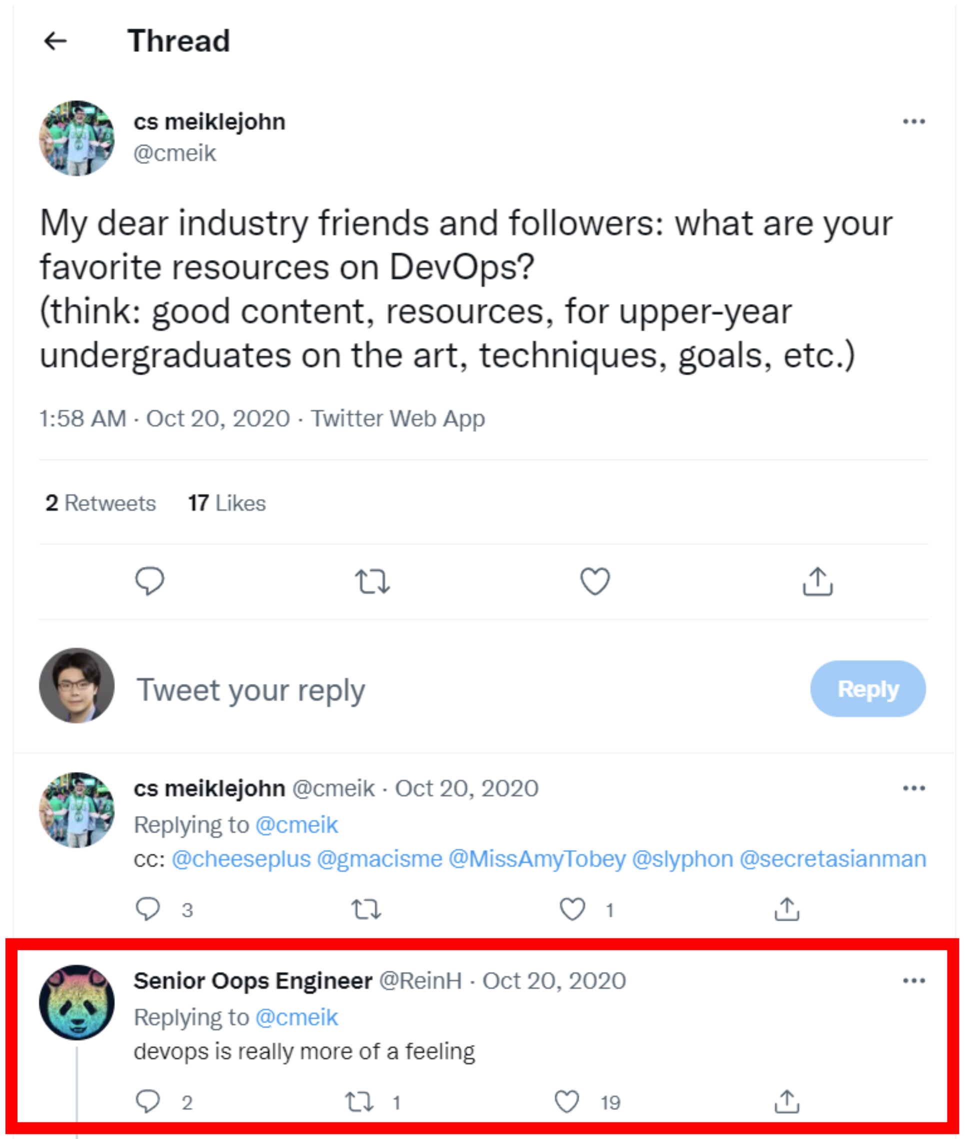
View all in chart

Top 5 Package	Betweenness
debug	61860.30
yeoman-generator	60182.34
readable-stream	49337.18
npm	48366.35
request	42349.79

View all in chart

https://graphcommons.com/graphs/a7ec343d-2a0c-47bb-9658-bb8315e8a096?auto=true&show=analysis-cluster

What is DevOps?



Thread

cs meiklejohn
@cmeik

My dear industry friends and followers: what are your favorite resources on DevOps?
(think: good content, resources, for upper-year undergraduates on the art, techniques, goals, etc.)

1:58 AM · Oct 20, 2020 · Twitter Web App

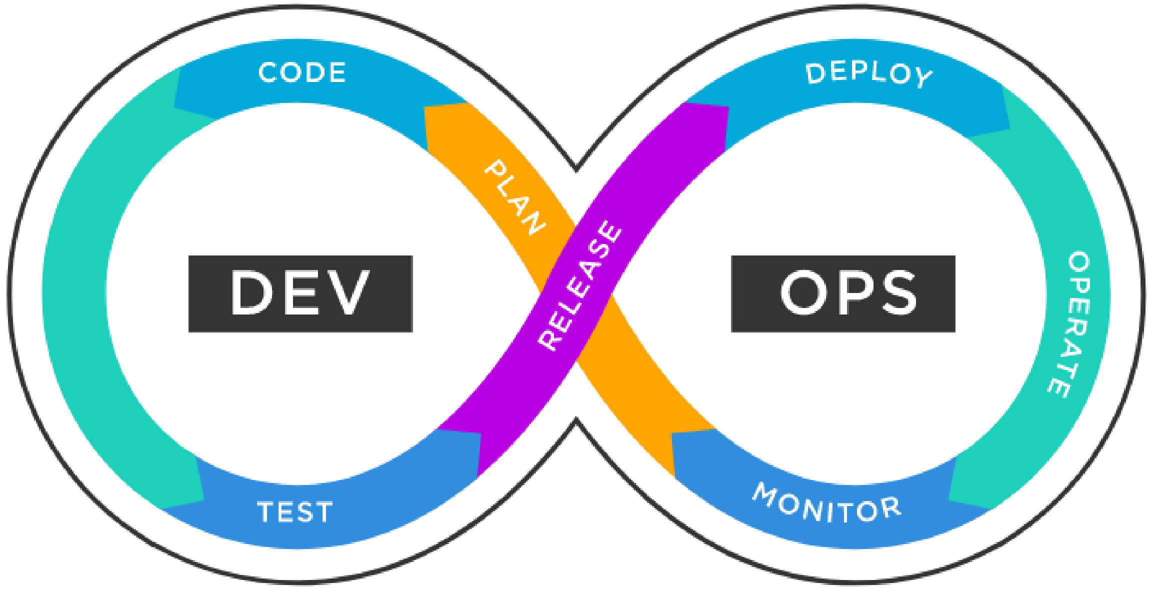
2 Retweets 17 Likes

Tweet your reply

cs meiklejohn @cmeik · Oct 20, 2020
Replying to @cmeik
cc: @cheeseplus @gmacisme @MissAmyTobey @slyphon @secretasianman

Senior Oops Engineer @ReinH · Oct 20, 2020
Replying to @cmeik
devops is really more of a feeling

- Some best practices for software **Dev**elopment and **Op**eration, mainly for efficiency and automation, that is settled down from decades of software practitioner experiences



https://www.tibco.com/reference-center/what-is-devops

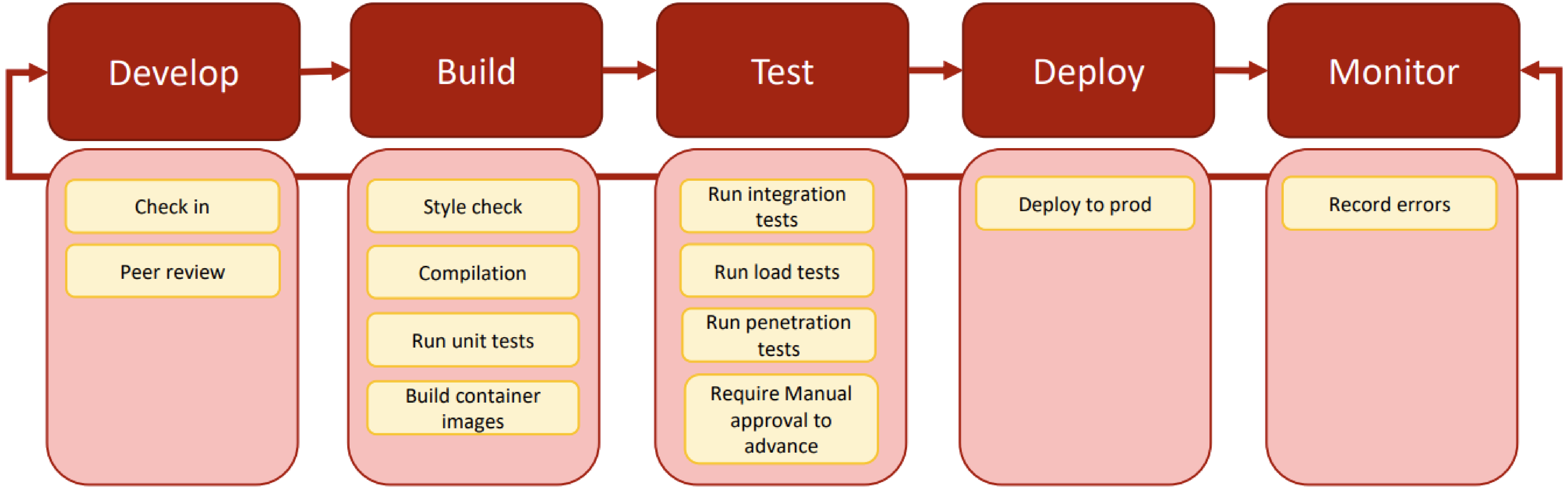
Key Components of DevOps

- Continuous Integration **(CI)**
  1. Constant testing as code is checked-in/pushed to the repository (e.g., GH hooks, etc.)
  2. Verify the build process works (i.e., parsing, compilation, code generation, etc.)
  3. Verify unit tests pass, style checks pass, other static analysis tools.
  4. Build artifacts
- Continuous Delivery & Deployment **(CD)**
  1. Moving build artifacts from test -> stage -> prod environments.
     Environments always differ! (e.g., ENV, PII, data, etc.)
  2. Gate code, if necessary, from advancing without manual approval.
     Useful when initially transitioning applications into a modern DevOps pipeline.

Key Components of DevOps

- Infrastructure as Code
  1. Required resources (e.g., cloud services, access policies, etc.) are created by code. No UI provisioning, no manual steps (avoid: easy to forget, time consuming!)
  2. “Immutable Infrastructure” No update-in-place (e.g., SSH to server.) Replace with new instances, decommission old instances.
  3. Nothing to prod without it being in code, checked-in, versioned along side code!
- Observability (Monitoring, Logging, Tracing, Metrics)
  1. Be able to know how your application is running in production
  2. Track and analyze low-level metrics on performance, resource allocation
  3. Capture high-level metrics on application behavior
     1. What’s “normal”? 2. What’s abnormal?

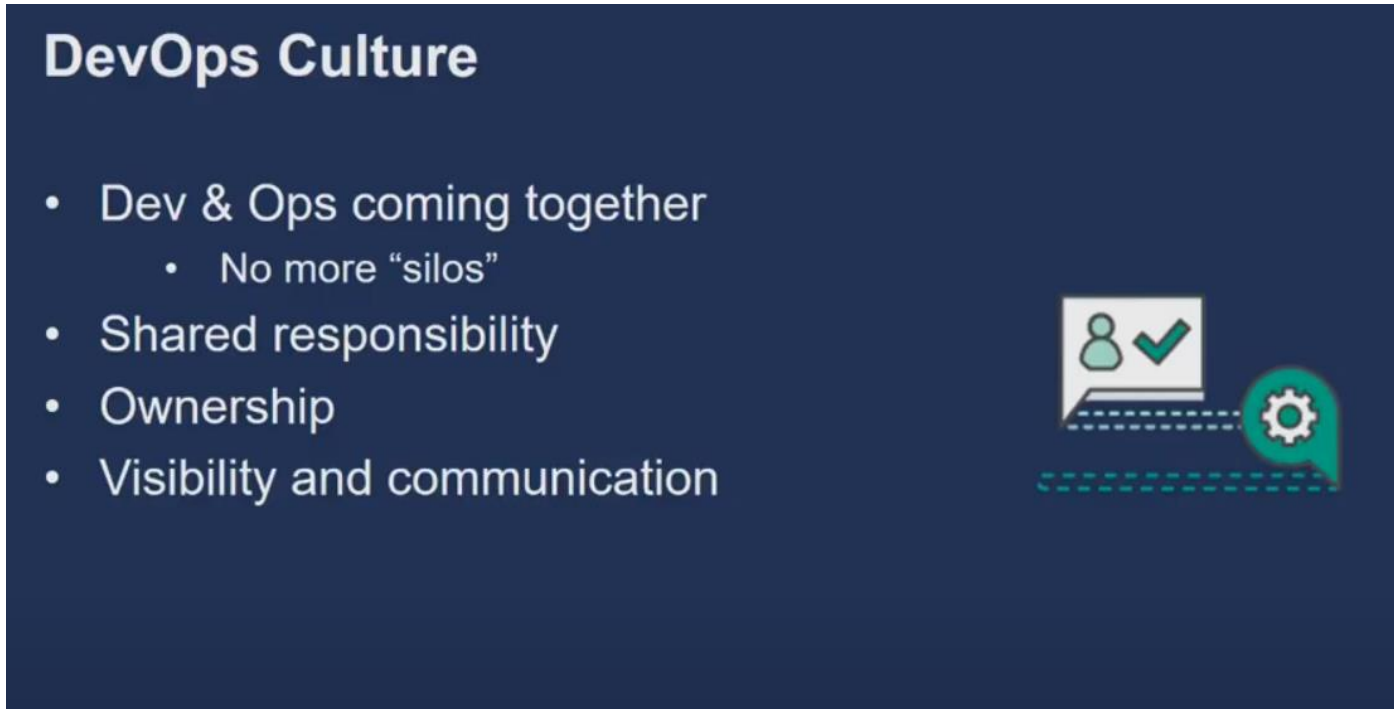
Typical CI & CD Pipelines



https://cmu-313.github.io/2020/lectures/15-Devops.pdf

DevOps is a **Culture**, Supported by Tools and Practices

- Build cohesive, multidisciplinary teams. Typically, developers are the “first responders” when things go bad in production. Sense of “ownership” by the developer all the way from inception to release.



DevOps Culture

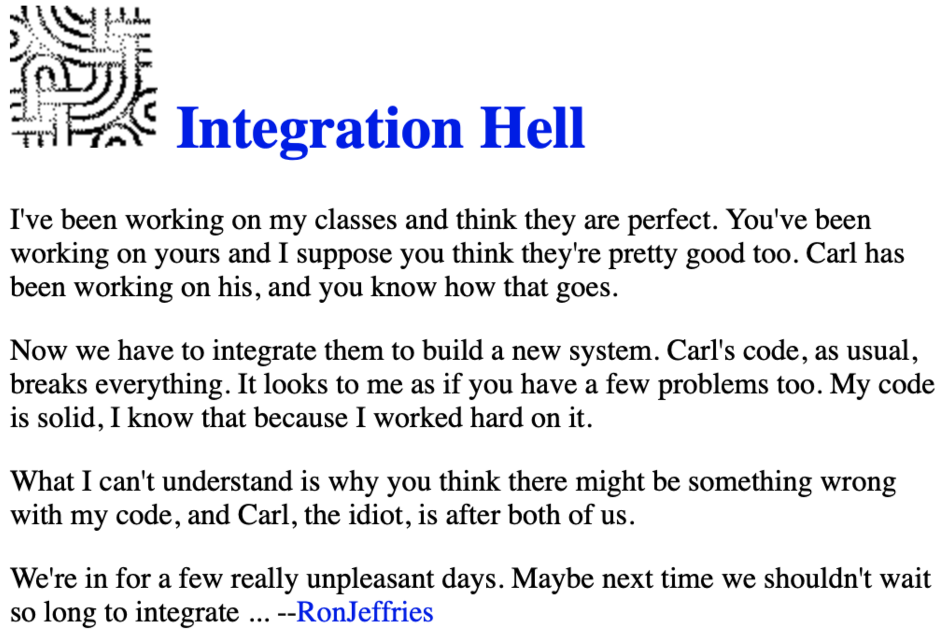
- Dev & Ops coming together
  - No more “silos”
- Shared responsibility
- Ownership
- Visibility and communication

https://www.youtube.com/watch?v=mBU3AJ3j1rg

What Happens Then?

- Small team => reduced frictions
- Automated pipelines => reduced waiting

➡ **Everything Goes Faster!**
And (Somehow) Safe and Steady



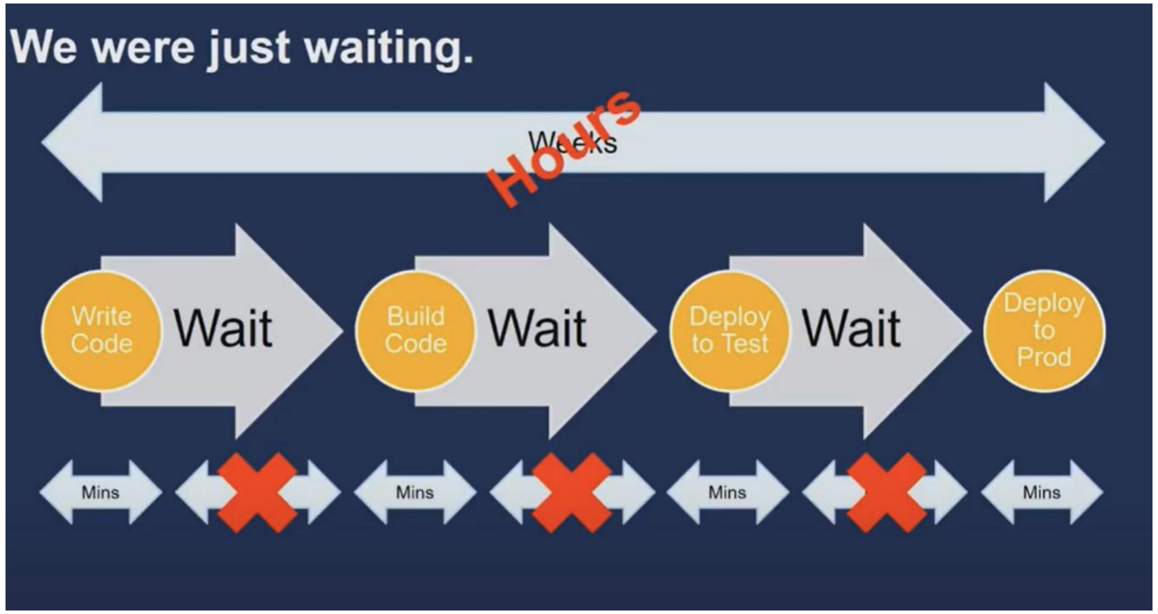
Integration Hell

I've been working on my classes and think they are perfect. You've been working on yours and I suppose you think they're pretty good too. Carl has been working on his, and you know how that goes.

Now we have to integrate them to build a new system. Carl's code, as usual, breaks everything. It looks to me as if you have a few problems too. My code is solid, I know that because I worked hard on it.

What I can't understand is why you think there might be something wrong with my code, and Carl, the idiot, is after both of us.

We're in for a few really unpleasant days. Maybe next time we shouldn't wait so long to integrate ... --RonJeffries

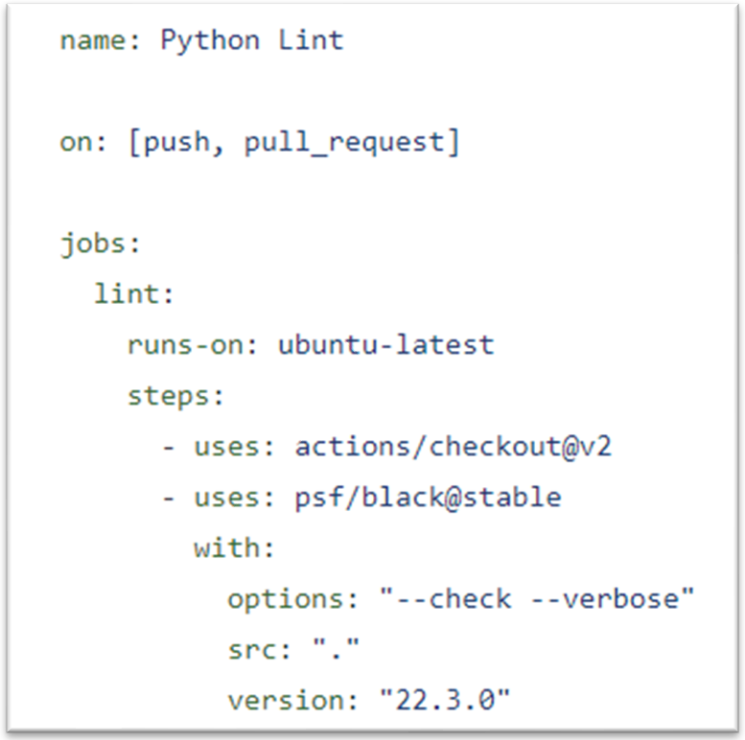


https://www.youtube.com/watch?v=mBU3AJ3j1rg

CI & CD Services

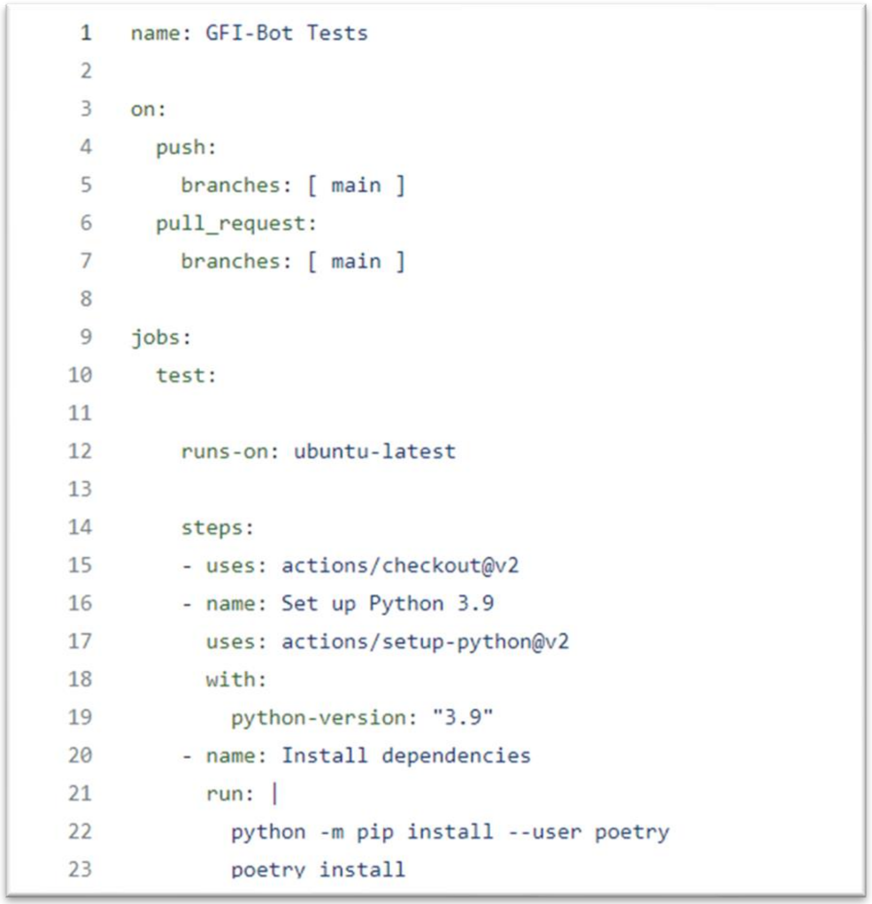
- GitHub Actions
  - 为GitHub仓库自定义工作流（构建、测试、打包、发布、部署等）的系统，通过.github/workflows目录下的YAML文件配置
  - https://docs.github.com/cn/actions
- GitLab CI/CD
  - 基于 GitLab 的 CI/CD 系统，通过 .gitlab-ci.yml 在项目中配置 CI/CD 流程
  - https://docs.gitlab.com/ee/ci/
- Gitee GO
  - Gitee 推出的 CI/CD(持续构建与集成)服务。用户可以通过自定义.workflow/中的YAML文件，实现构建集成自动化
- Travis CI
  - Travis CI是第三方持续集成服务，通过自定义配置文件 .travis.yml，构建和测试托管在GitHub的软件项目
  - https://docs.travis-ci.com
- Jenkins
  - Jenkins是一个开源的、提供友好操作界面的CI工具，支持多种版本控制系统，具有丰富的插件支持
  - https://www.jenkins.io/doc/

Example CI & CD Pipeline in GitHub Action

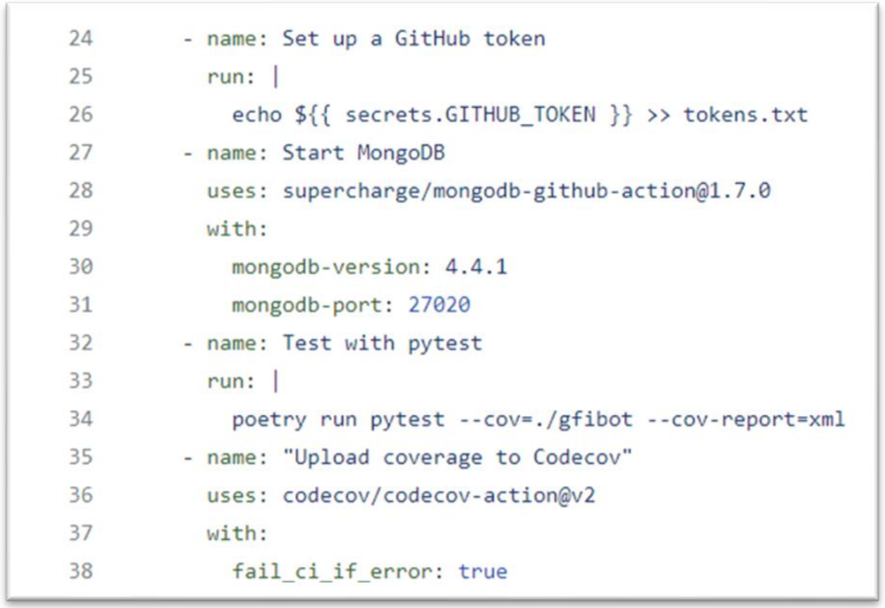


```yaml
name: Python Lint

on: [push, pull_request]

jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: psf/black@stable
        with:
          options: "--check --verbose"
          src: "."
          version: "22.3.0"
```

Static Analysis



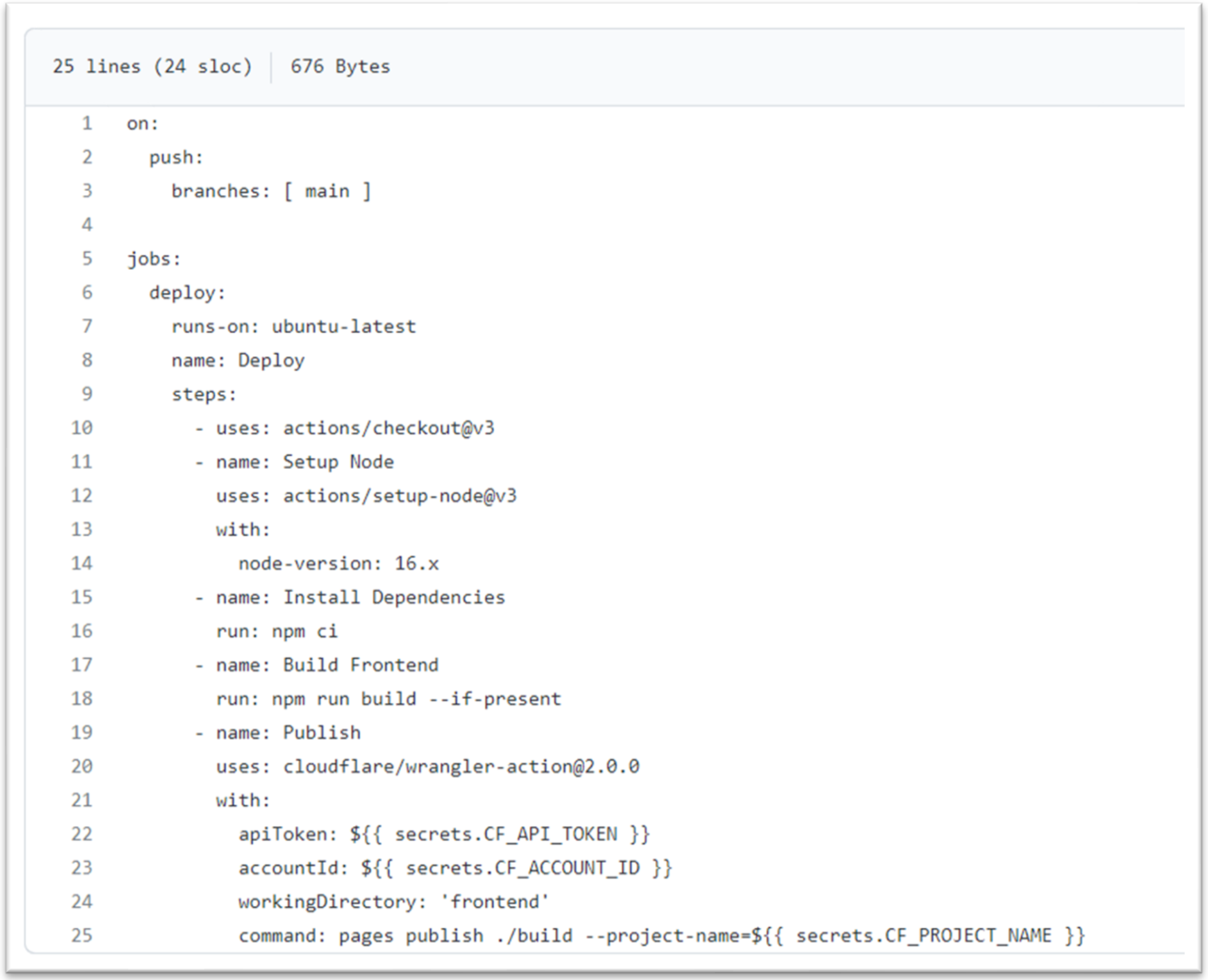
```yaml
name: GFI-Bot Tests

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  test:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python 3.9
      uses: actions/setup-python@v2
      with:
        python-version: "3.9"
    - name: Install dependencies
      run: |
        python -m pip install --user poetry
        poetry install
```



```yaml
    - name: Set up a GitHub token
      run: |
        echo ${{ secrets.GITHUB_TOKEN }} >> tokens.txt
    - name: Start MongoDB
      uses: supercharge/mongodb-github-action@1.7.0
      with:
        mongodb-version: 4.4.1
        mongodb-port: 27020
    - name: Test with pytest
      run: |
        poetry run pytest --cov=./gfibot --cov-report=xml
    - name: "Upload coverage to Codecov"
      uses: codecov/codecov-action@v2
      with:
        fail_ci_if_error: true
```

Describe Dev Environment + Run Tests

Example CI & CD Pipeline in GitHub Action

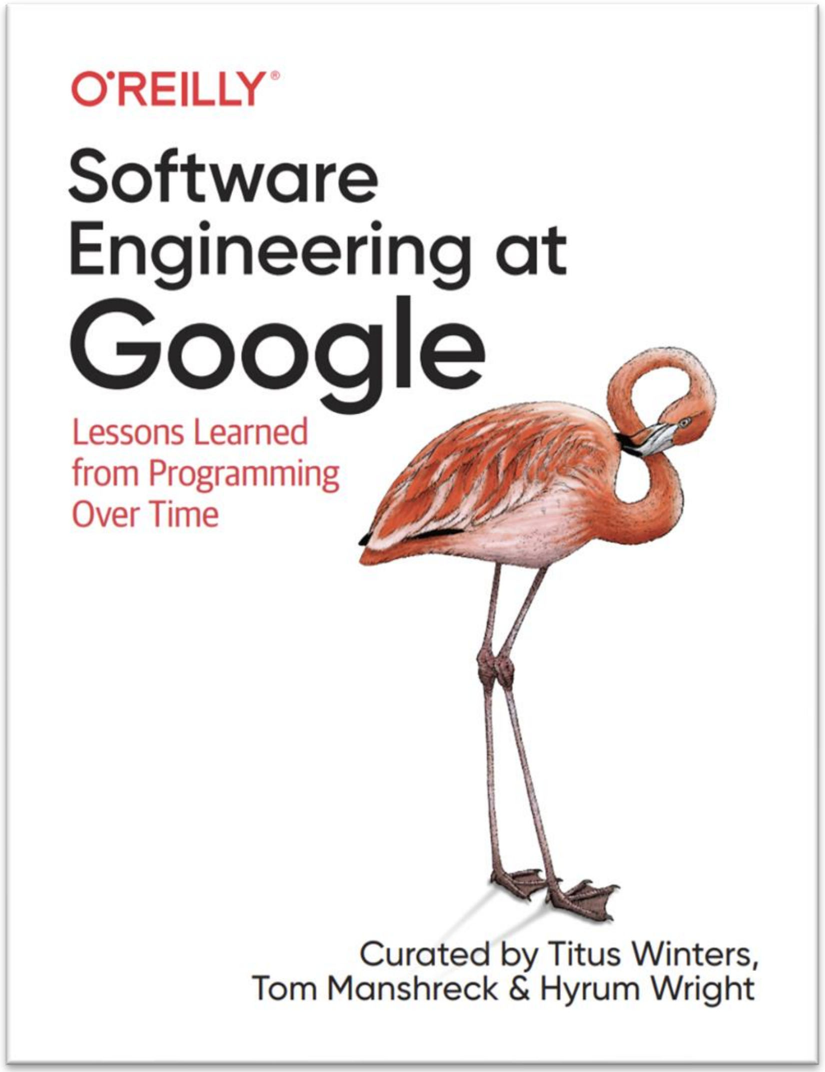


```
25 lines (24 sloc) | 676 Bytes
```

```yaml
on:
  push:
    branches: [ main ]

jobs:
  deploy:
    runs-on: ubuntu-latest
    name: Deploy
    steps:
      - uses: actions/checkout@v3
      - name: Setup Node
        uses: actions/setup-node@v3
        with:
          node-version: 16.x
      - name: Install Dependencies
        run: npm ci
      - name: Build Frontend
        run: npm run build --if-present
      - name: Publish
        uses: cloudflare/wrangler-action@2.0.0
        with:
          apiToken: ${{ secrets.CF_API_TOKEN }}
          accountId: ${{ secrets.CF_ACCOUNT_ID }}
          workingDirectory: 'frontend'
          command: pages publish ./build --project-name=${{ secrets.CF_PROJECT_NAME }}
```

Automated Deployment

Suggested Reading


O'REILLY
Software Engineering at Google
Lessons Learned from Programming Over Time
Curated by Titus Winters, Tom Manshreck & Hyrum Wright

Some Additional References

- CMU 15-313 Foundations of Software Engineering
  - https://cmu-313.github.io/2020/
  - https://cmu-313.github.io/2020/lectures/15-Devops.pdf
- DevOps at Amazon https://www.youtube.com/watch?v=mBU3AJ3j1rg
- DevOps at Netflix https://www.youtube.com/watch?v=UTKIT6STSVM
- DevOps at IBM https://www.youtube.com/watch?v=UbtB4sMaaNM
- GitHub Action https://docs.github.com/en/actions
- Docker: https://www.docker.com/get-started/
- Software Engineering at Google
  - https://abseil.io/resources/swe-book/html/toc.html

Lab 4: CI & CD for a Python Package

- 为一个简单的Python包pygraph
  - 配置Python开发环境
  - 配置Pre-Commit Hook
  - 实现一些简单功能并通过测试
  - 配置五阶段CI/CD流水线
    - 初始化Python环境，安装Poetry
    - 使用Poetry自动安装所有依赖
    - 使用black检测代码是否存在格式问题
    - 使用pytest运行单元测试
    - 使用pdoc3生成API文档，并将API文档部署到仓库中的gh-page分支（部署Python包的部分留到下个lab）
- https://github.com/osslab-pku/OSSDevelopment/blob/main/Assignments/Lab4.md