



北京大学

# Lab 4: CI & CD & DevOps

徐卫伟

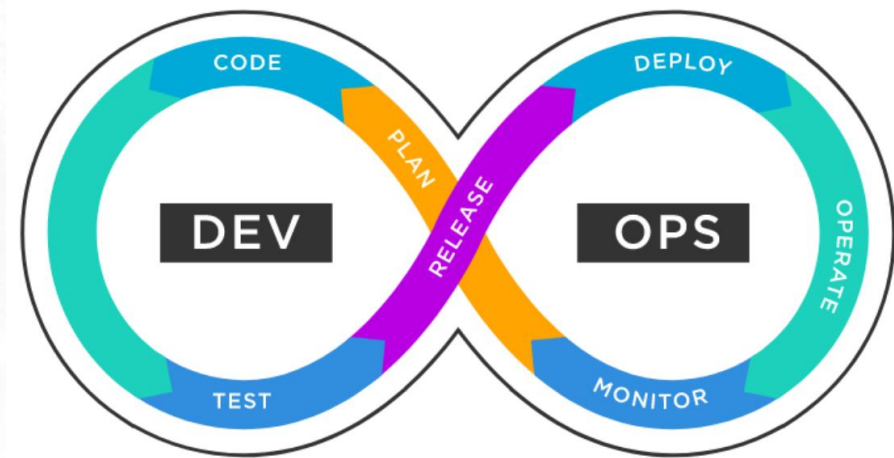
2024. 10. 30

# Definition of DevOps



What is DevOps?

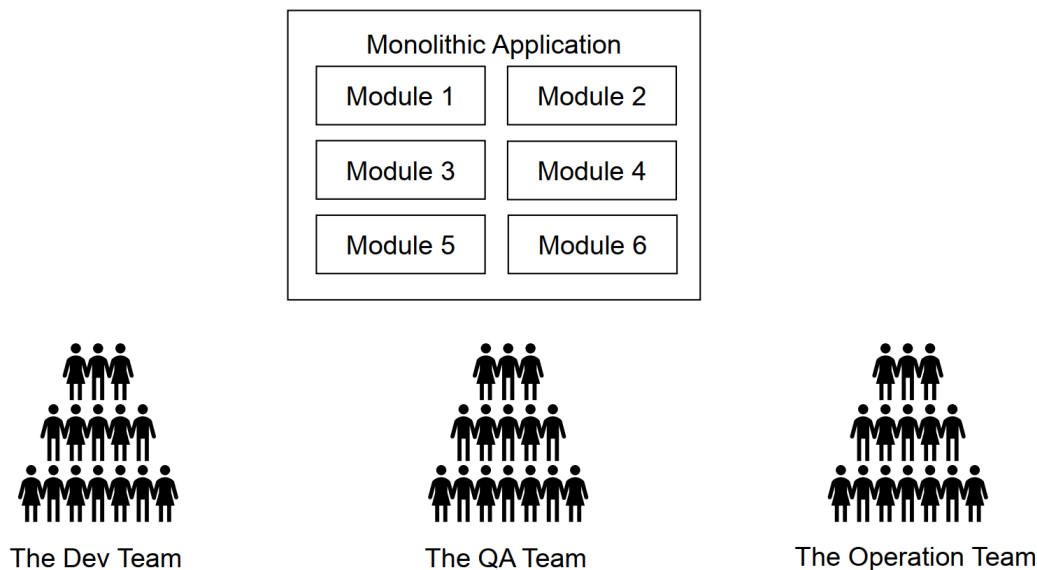
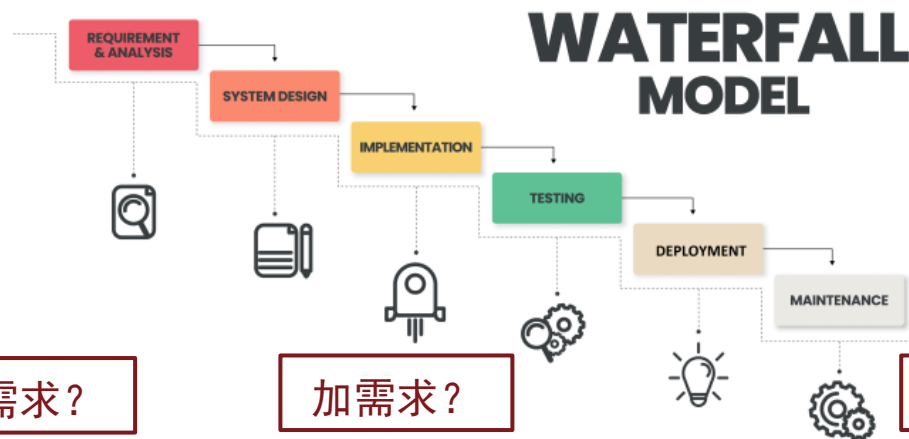
- DevOps combines development (Dev) and operations (Ops) to unite **people, process, and technology** in application planning, development, delivery, and operations. DevOps enables coordination and collaboration between formerly siloed roles like development, IT operations, quality engineering, and security.
- Teams adopt DevOps culture, practices, and tools to increase confidence in the applications they build, respond better to customer needs, and achieve business goals faster. DevOps helps teams continually provide **value** to customers by producing better, more reliable products.



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

<https://learn.microsoft.com/en-us/devops/what-is-devops>

# Before DevOps: The Traditional Software Development



Problem: Changes are troublesome.



## Integration Hell

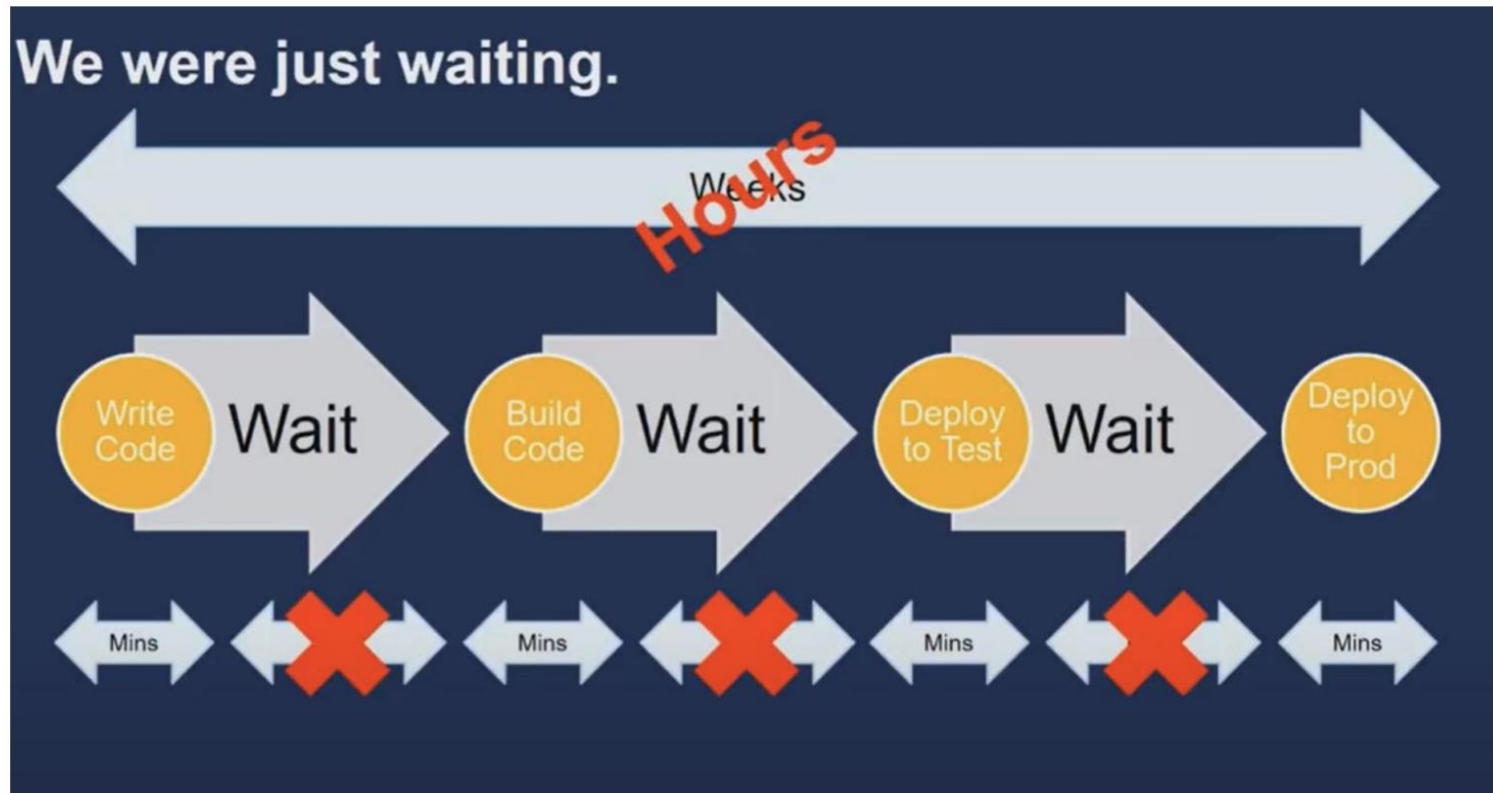
I've been working on my classes and think they are perfect. You've been working on yours and I suppose you think they're pretty good too. Carl has been working on his, and you know how that goes.

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.

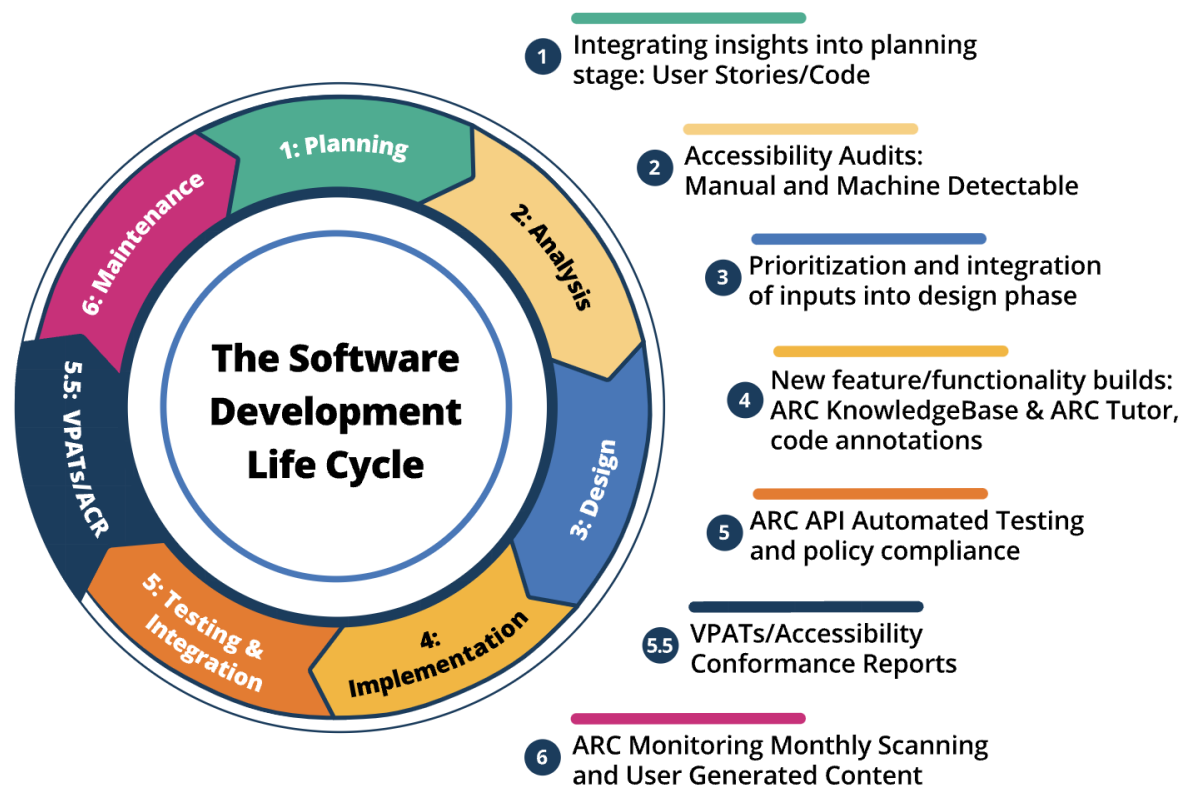
What I can't understand is why you think there might be something wrong with my code, and Carl, the idiot, is after both of us.

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

# Before DevOps: The Traditional Software Development



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



<https://www.tpgi.com/how-to-efficiently-include-accessibility-testing-through-continuous-integration/>



## Development transformation at Amazon: 2001-2009

2001



Monolithic  
application + teams

2009



Microservices + 2 pizza teams

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



## Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
Customer collaboration over contract negotiation  
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

## 敏捷开发宣言

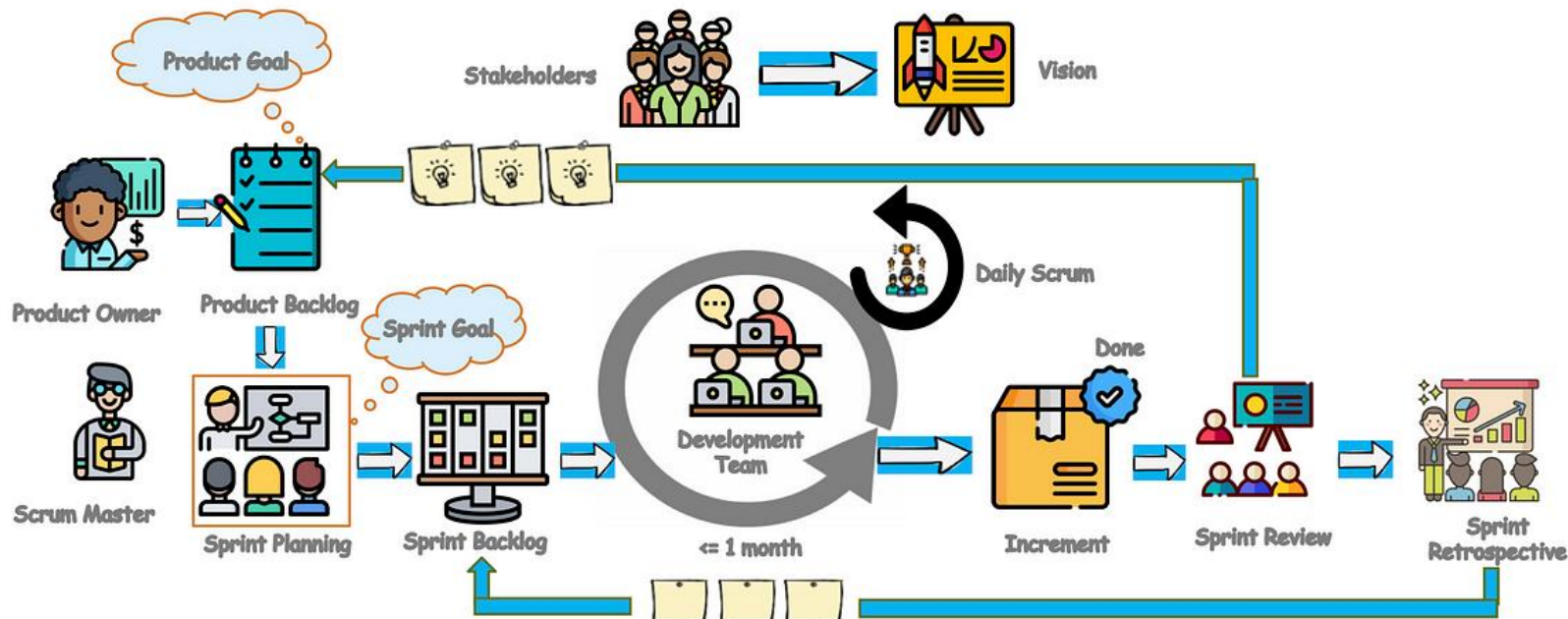
<https://agilemanifesto.org/iso/zhchs/manifesto.html>



# Agile—Scrum

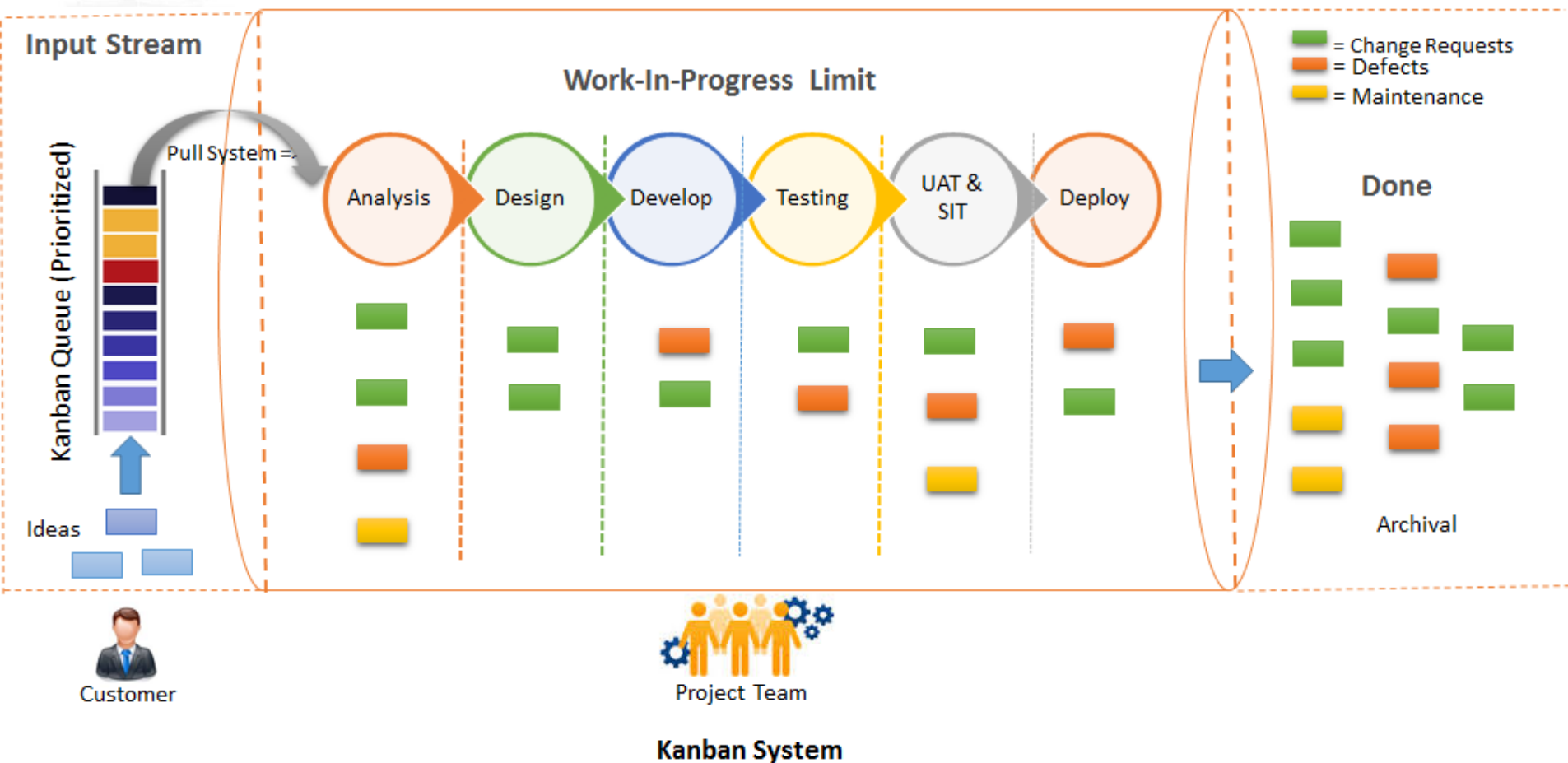


北京大学  
PEKING UNIVERSITY

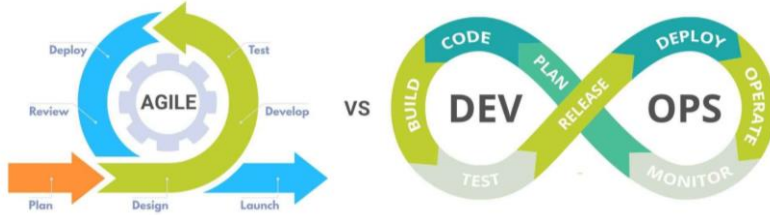


<https://medium.com/agilemania/essential-elements-of-agile-scrum-d5d5d8cafe8d>

# Agile——Kanban(かんばん,看板)



# DevOps vs. Agile



When the agile methodology gained widespread adoption in the early 2000s, it transformed the way we develop software and other products. Yet, within a few years of becoming an industry standard, a critical oversight arose: **the processes and requirements of the operations** team who deployed and managed software products were left out of the revolution.

This led to DevOps, an approach that aligned development and operations teams.





## DEVOPS VS. AGILE



DevOps	Parameter	Agile Methodology
Emphasis on collaboration & productivity	Philosophy 	Emphasis on incremental changes through iterative development & testing
Continuous testing & integration, providing end-to-end business solutions	Focal Point & Purpose 	Incremental deployments in complex projects
Looks after secure deployment	Delivery & Deployment 	Looks after developing & launching software
Large team with different skill-set	Team Size & Skills 	Smaller team with advanced skill-sets
Extensive documentation	Documentation 	Light on documentation
Received via customers	Feedback 	Received internally
Specs & design documents	Communication 	Daily scrum meetings

## Waterfall vs Agile vs DevOps

demandsimplified.marketing

	Waterfall 	Agile 	DevOps 
Approach	Waterfall model provides a linear sequential approach to managing software projects. Each phase depends on deliverables from the previous one.	Agile is a highly dynamic and iterative approach where you do not need the complete set of requirements to start with. You can develop some features and check customer response before next steps	DevOps is an Agile methodology encompassing Development (Dev) and Operations (Ops). It enables end-to-end lifecycle delivery of features, fixes, and updates at frequent intervals.
Year	1970	2001	2009
Scope and Schedule	Adjust schedule to preserve scope. Fixed requirements. Limited flexibility.	Adjust scope to preserve schedule. Iterative approach allows for prioritization	Adjust scope to preserve schedule. Highly responsive to business needs
Time to Market	Slow	Rapid	Continuous
Automation	Low	Varied	High
Customer Feedback	End of Project	After every sprint	Continuous
Quality	Low. Issues are identified only at testing stage	Better. Issues are identified after every sprint	High. Automated unit testing during development
Collaboration	Teams operate in silos	Multiple teams are involved, but not all	All stakeholders are involved from start to finish
Variations	V-model	Scrum, XP, LeSS, SAFe	CI/CD, DevSecOps

<https://www.linkedin.com/pulse/waterfall-vs-agile-devops-sdlc-models-abhay-reddy>

- **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.

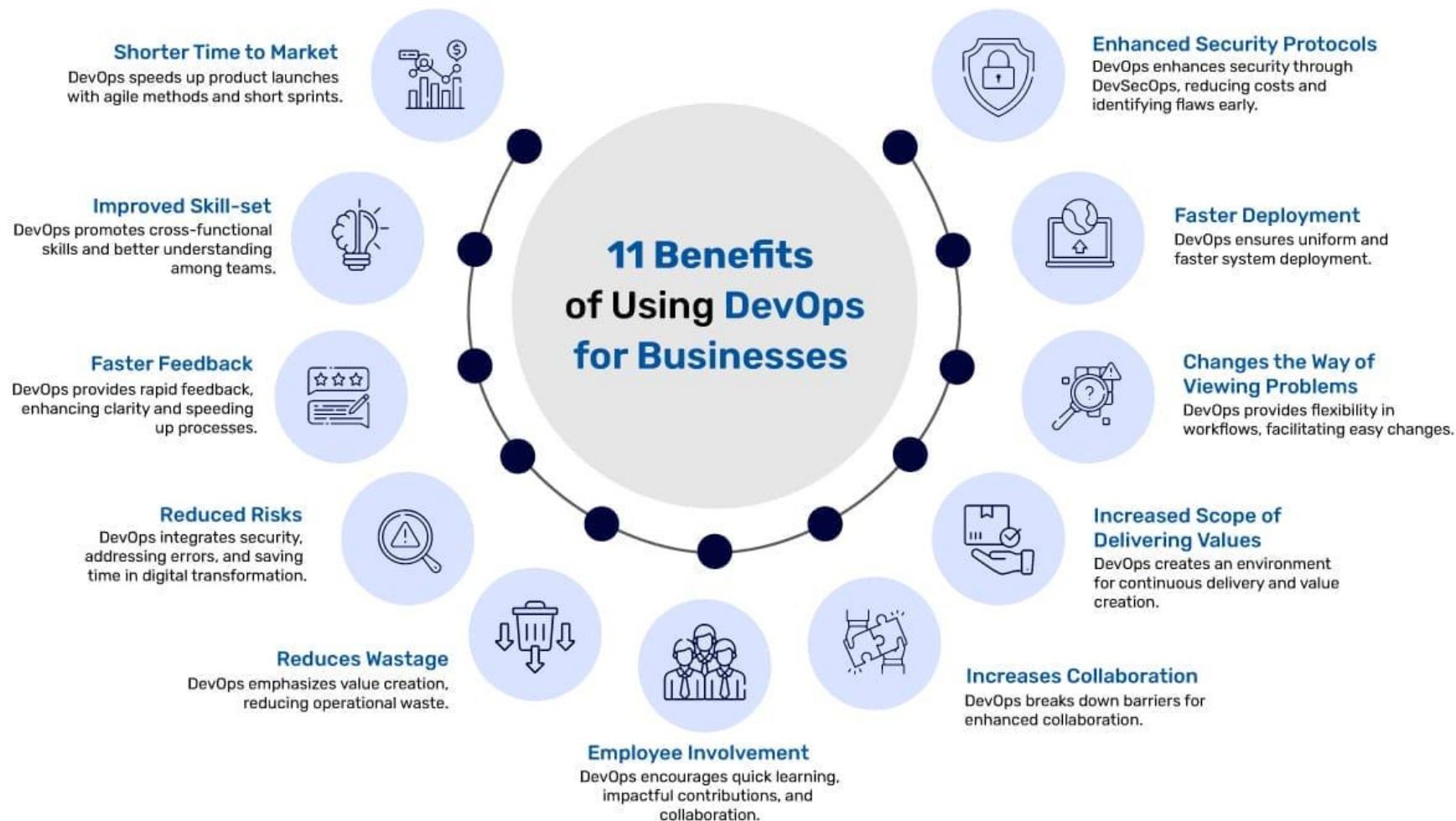
- **Infrastructure as Code**

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)**

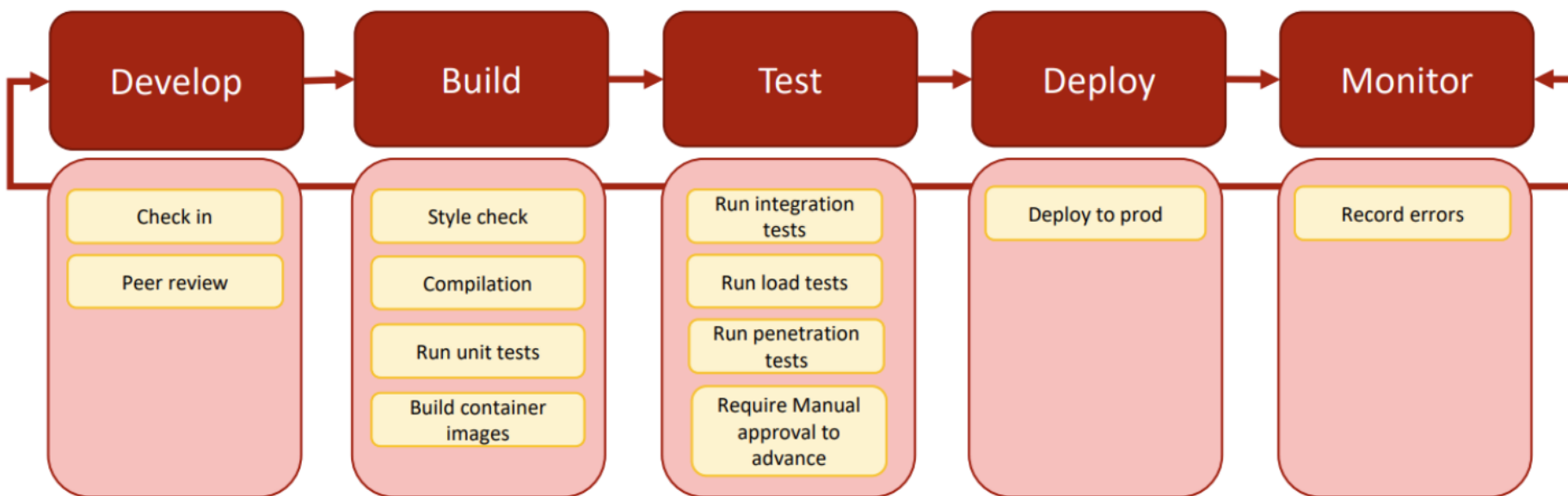
1. Be able to know how your application is running in production
2. Track and analyze low-level metrics on performance, resource allocation
3. Capture high-level metrics on application behavior
  1. What's “normal”? 2. What's abnormal?





<https://www.mindbrowser.com/benefits-of-devops-for-business/>

# Typical CI & CD Pipelines



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

- 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/>

# GitHub Actions Example



```
yml
1 name: CI/CD Pipeline
2
3 # 定义触发工作流的事件
4 on:
5   push:
6     branches:
7       - main # 监听推送到 main 分支的事件
8
9 # 定义作业
10 jobs:
11   build-and-test:
12     # 指定运行环境
13     runs-on: ubuntu-latest
14
15     # 定义工作步骤
16     steps:
17       - name: Check out the code
18         uses: actions/checkout@v2 # 检出最新的代码版本
19
20       - name: Set up Python
21         uses: actions/setup-python@v2 # 配置 Python 环境
22         with:
23           python-version: '3.8'
24
25       - name: Install dependencies
26         run: |
27           python -m pip install --upgrade pip
28           pip install -r requirements.txt
29
30       - name: Run tests
31         run: |
32           pytest # 运行测试
33
34   deploy:
35     needs: build-and-test # 依赖另一个作业的结果
36     runs-on: ubuntu-latest
37
38     steps:
39       - name: Deploy to server
40         run: echo "Deploying to production server..."
41         # 部署到生产服务器的具体命令或操作
```

## 基本语法

- **name :**  
工作流的名称。此字段是可选的，但有助于在 GitHub 界面中清晰地标识工作流。
- **on :**  
定义触发工作流的事件。例如，push、pull\_request、schedule 等。
- **jobs :**  
工作流中的一组任务。每个作业都可以在不同的环境中执行，具有独立的运行上下文。
- **runs-on :**  
定义作业运行的虚拟环境，如 ubuntu-latest、macos-latest 或 windows-latest。
- **steps :**  
作业的执行步骤，可以是单行命令或调用特定的 GitHub Actions。
- **uses :**  
用于指定步骤中调用的动作，这可以是官方或社区提供的动作库。
- **with :**  
用于在 uses 中传递参数到操作。

# Some Additional References



- CMU 15-313 Foundations of Software Engineering  
<https://cmu-313.github.io/2020/>
- 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>