

CS 4530 & CS 5500

Software Engineering

Lecture 10.2: Continuous Integration

Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences
© 2021, released under [CC BY-SA](#)

Learning Objectives for this Lesson

By the end of this lesson, you should be able to...

- Describe how continuous integration helps to catch errors sooner in the software lifecycle
- Use continuous integration systems to automate testing in real software projects

Cost to Fix a Defect Over Time

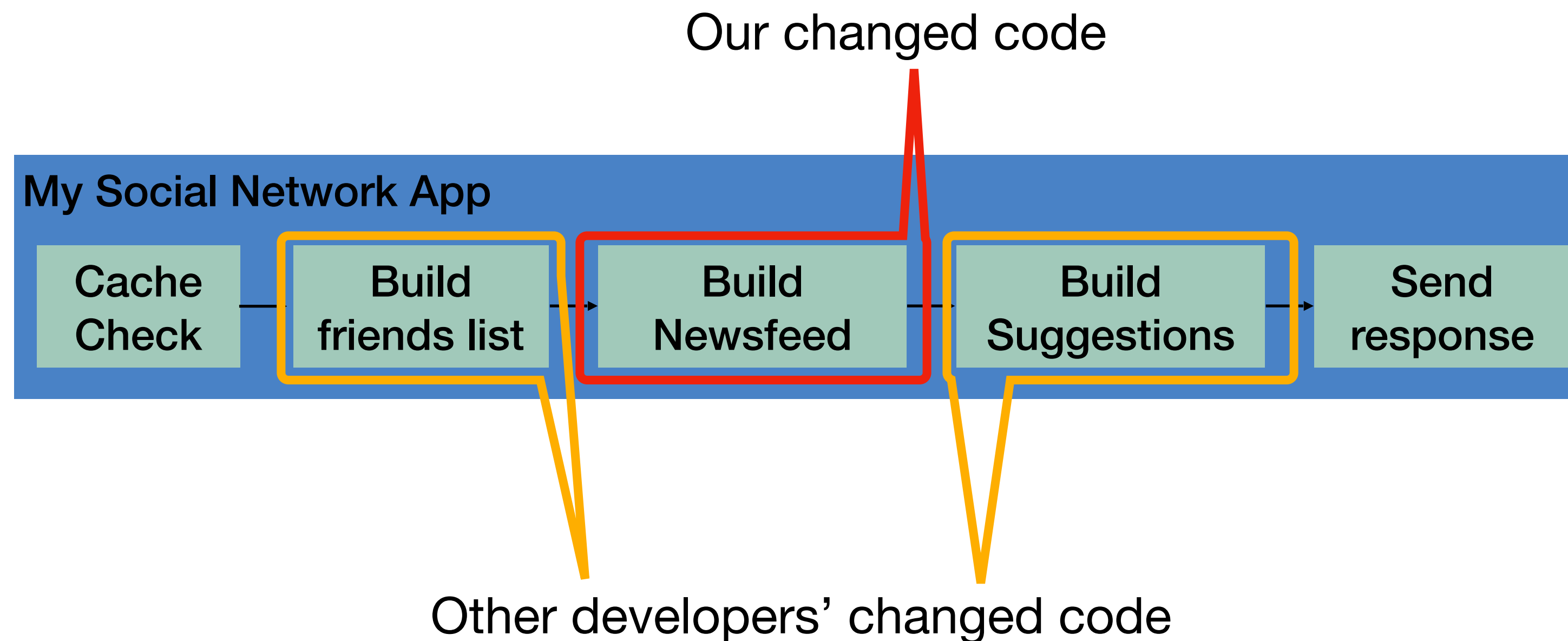
Rough estimate



Continuous Integration

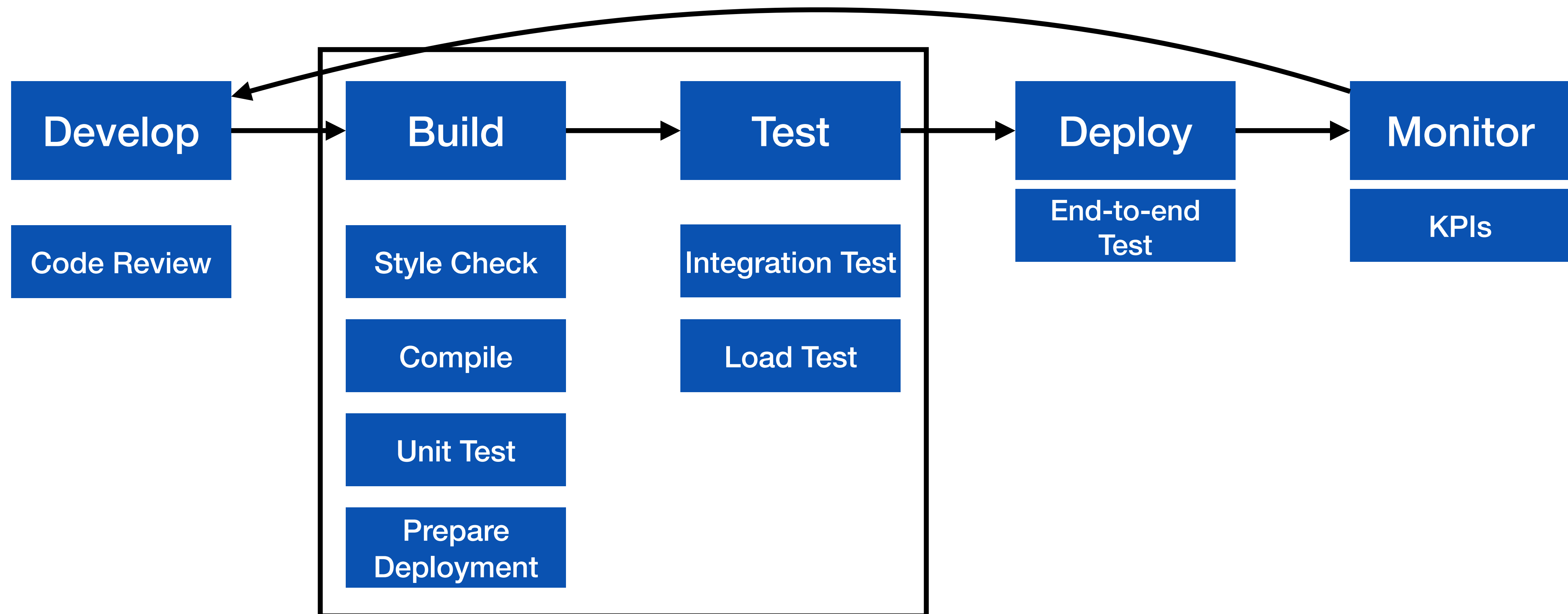
Motivation

- Our systems involve many components, some of which might even be in different version control repositories
- How does a developer get feedback on their (local) change?



Continuous Integration

Continuously assembling and testing our entire codebase



Automate this centrally, provide a central record of results

Build Systems

Automatically compiling code and generating executables

- You've probably used multiple of these:
 - Make, maven, ant, gradle, grunt, sbt
- Why use a build system?
 - Builds should be repeatable
 - Builds should be reproducible
 - Builds should be standard

Build Systems

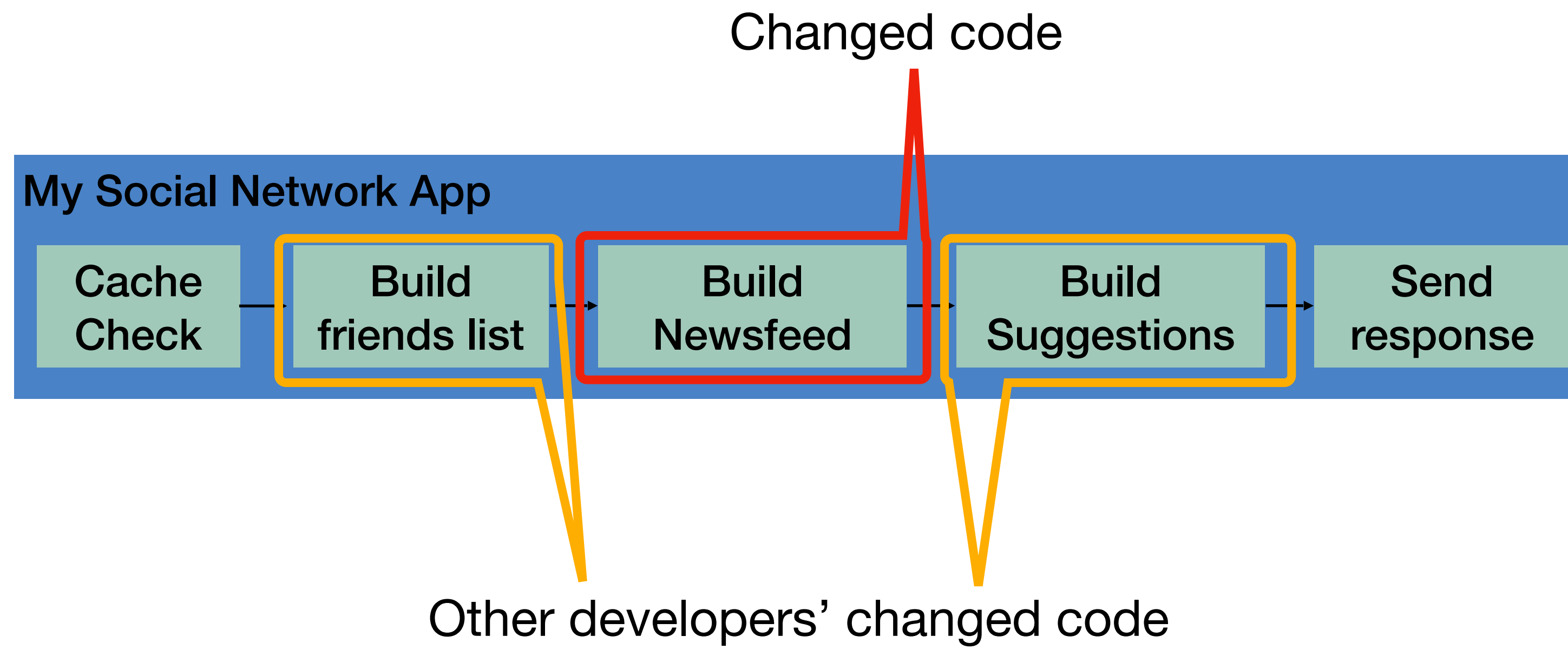
Not just compilation

- Fetch dependencies and link them (using a package manager like maven, pip or npm)
- Provision & teardown resources for integration testing
- Run tests
- Generate a release archive
- Ideally, do this all in parallel as much as possible

How do we apply continuous integration?

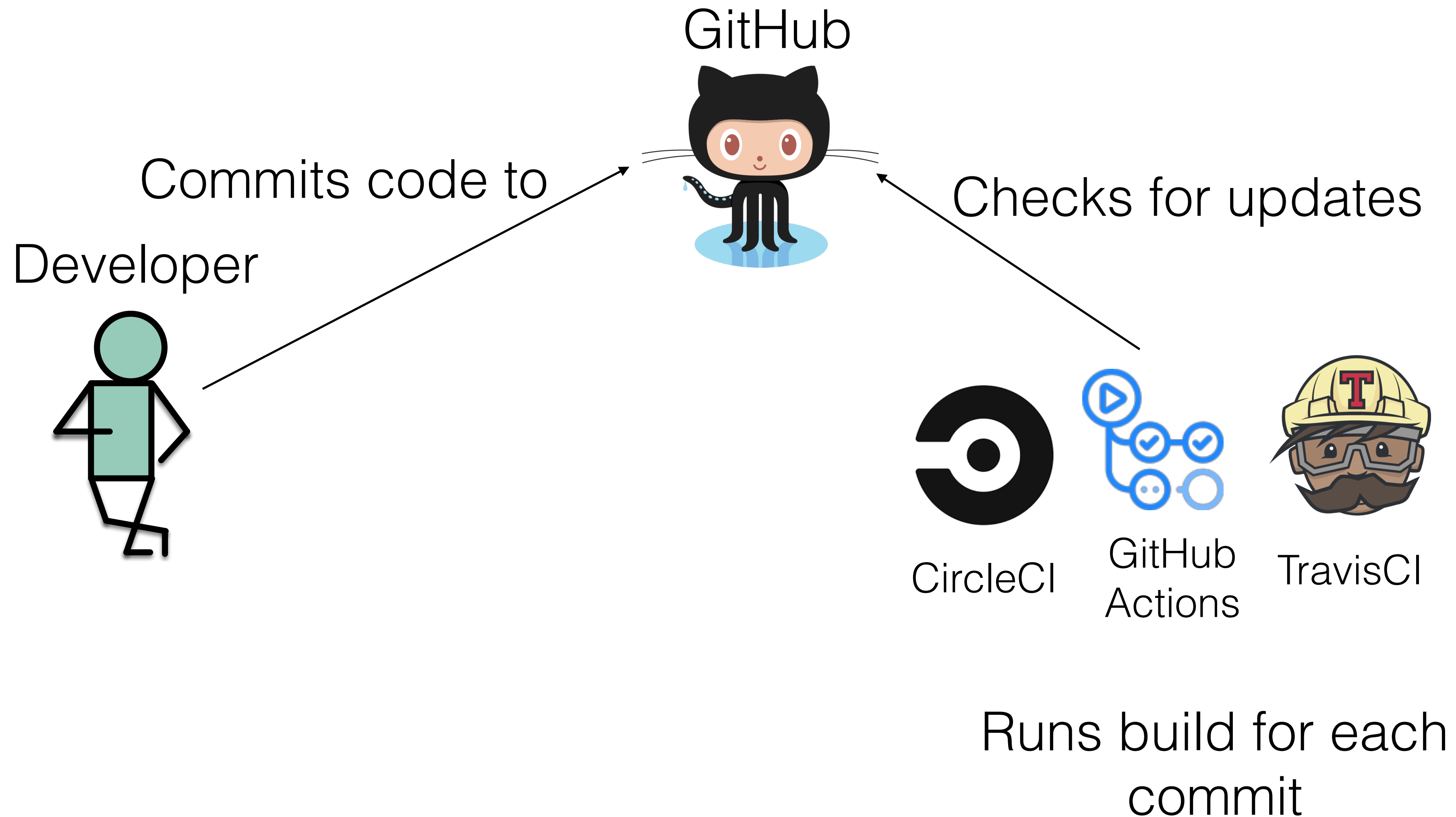
Testing the right things at the right time

- Do we integrate changes immediately, or do a pre-commit test?
- Which tests do we run when we integrate?
- How do we compose the system under test at each point?



Continuous Integration in Practice

Small scale, with a service like CircleCI, GitHub Actions or TravisCI



Continuous Integration in Practice

Large scale example: Google TAP

- >50,000 unique changes per-day, > 4 billion test cases per-day
- Pre-submit optimization: run fast tests for each individual change (before code review). If fast tests pass, allow the merge to continue
- Then: run all affected tests; “build cop” monitors and acts immediately to roll-back or fix
- Build cop monitors integration test runs
- Average wait time to submit a change: 11 minutes

Continuous Integration in Practice

Medium-scale example with branches

Git Branch	Continuous Integration Pipeline			
Develop	lint	unit test		
Staging	lint	unit test	integration test	
Stable	lint	unit test	integration test	deploy

Example CI Pipeline

Open source project: PrestoDB

prestodb / presto  build passing

Current Branches Build History Pull Requests

More options 

✗ Pull Request #15372 Fix extracting logic in dynamic filtering when

When integrating with filter pushdown, we extract dynamic filter

Commit cde9e65 

#15372: Fix extracting logic in dynamic filtering when integrated with

Branch master 

 Ke

🔗 #52304 failed

Ran for 17 min 40 sec

Total time 10 hrs 26 min 10 sec

 10 hours ago

Build jobs

View config

✗ # 52304.1	AMD64	Trusty	</> Java	MAVEN_CHECKS=true	10 min 51 sec
✓ # 52304.2	AMD64	Trusty	</> Java	WEBUI_CHECKS=true	58 sec
✓ # 52304.3	AMD64	Trusty	</> Java	TEST_SPECIFIC_MODULES=presto-tests TE	6 min 7 sec
✓ # 52304.4	AMD64	Trusty	</> Java	TEST_SPECIFIC_MODULES=presto-tests TE	24 min 50 sec
✓ # 52304.5	AMD64	Trusty	</> Java	TEST_SPECIFIC_MODULES=presto-tests TE	7 min 45 sec
✓ # 52304.6	AMD64	Trusty	</> Java	TEST_SPECIFIC_MODULES=presto-tests TE	8 min 4 sec











































Example CI Pipeline - TravisCI

At a glance, see history of build

prestodb / presto  build passing

Current Branches Build History Pull Requests

More options 

 master  James Sun	This patch bumps Alluxio dependency to 2.3.0-2	 #52300 passed  36392a2 	 10 hrs 49 min 31 sec  2 days ago
 master  Andrii Rosa	Handle query level timeouts in Presto on Spark	 #52287 errored  aa55ea7 	 11 hrs 6 min 44 sec  2 days ago
 master  Wenlei Xie	Fix flaky test for TestTempStorageSingleStreamSp	 #52284 errored  193a4cd 	 11 hrs 50 min 37 sec  2 days ago
 master  Andrii Rosa	Check requirements under try-catch	 #52283 passed  fff331f 	 11 hrs 3 min 20 sec  2 days ago
 master  Maria Basmanova	Update TestHiveExternalWorkersQueries to create	 #52282 passed  746d7b5 	 10 hrs 55 min 37 sec  2 days ago
 master  Maria Basmanova	Introduce large dictionary mode in SliceDictionar	 #52277 passed  a89d97a 	 10 hrs 43 min 30 sec  2 days ago

<https://travis-ci.com/github/prestodb/presto>

Continuous Integration

Summary and next steps

- CI helps catch errors sooner in the software lifecycle by performing integration and end-to-end tests sooner
- CI can be applied in small-scale projects by running complete test suites for each commit, or in larger projects by running pre-commit tests per-commit and complete integrations regularly
- CI assumes the ability to automatically provision infrastructure on which to run those integration tests [next lesson]

This work is licensed under a Creative Commons Attribution-ShareAlike license

- This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>
- You are free to:
 - Share — copy and redistribute the material in any medium or format
 - Adapt — remix, transform, and build upon the material
 - for any purpose, even commercially.
- Under the following terms:
 - Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
 - ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
 - No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.