

Continuous Integration

CSCI 420: Software Engineering

Millersville University

Outline

- Definition
- Workflows
 - Locally Test
 - Remotely Build
 - Remotely Test
 - Remotely Deploy
- Steps to Success
- CI Services

Continuous Integration

The practice of merging all developers' working copies to a shared mainline several times a day

- Originated from “*Object-Oriented Design: With Applications*” by Grady Booch in 1991
- **Extreme Programming (XP)** adopted Continuous Integration as we know it today (late 1990s)

Continuous Integration: the Journey

Remotely Deploy

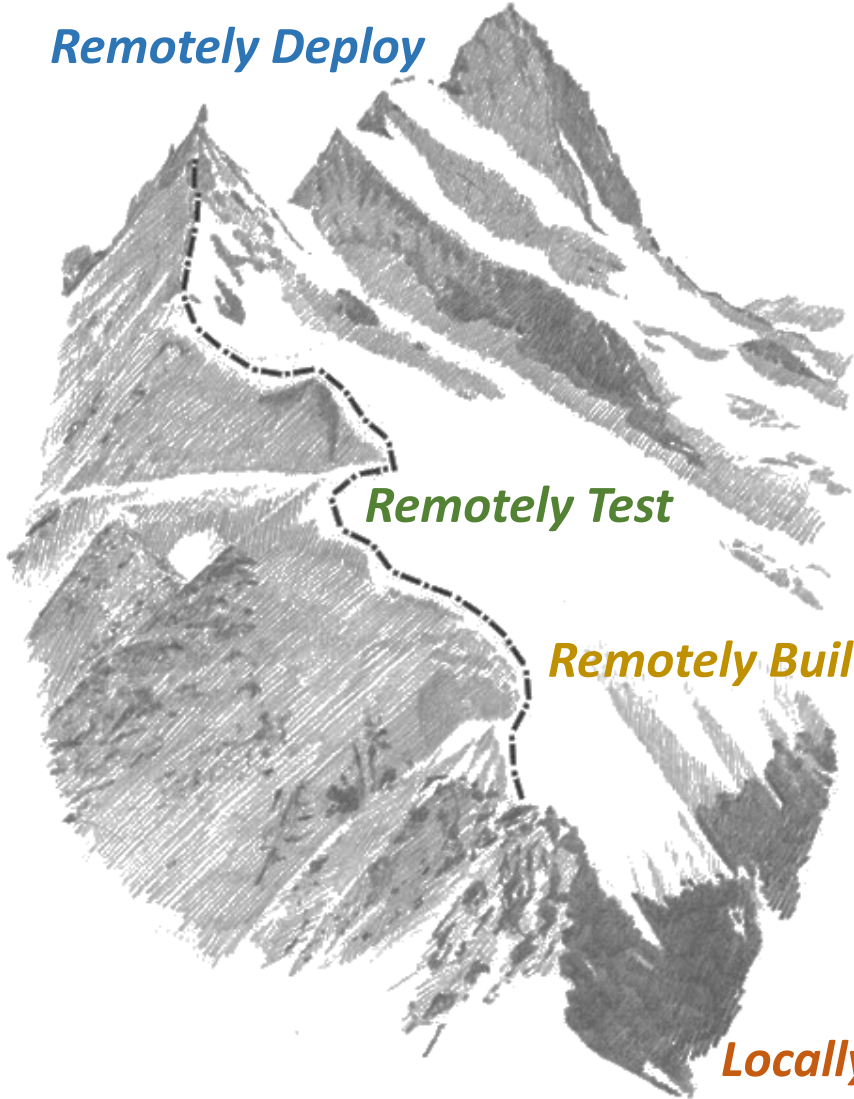
Remotely Test

Remotely Build

Locally Test

Locally Test

- Build and run tests on their own device
- Commit code to central repository
- Review other developers' pull requests, **trusting** the code works
 - Manual inspection of code
 - Manual inspection of tests
 - Still could be bugs



Continuous Integration: the Journey

Remotely Deploy

Remotely Test

Remotely Build

Locally Test

Remotely Build

- Build and run tests on their own device
- When code is committed to central repository, automatically build the project.
 - **If errors are reported, then the code cannot be merged**
- Review other developers' pull requests, **trusting** the code works
 - Manual inspection of code
 - Manual inspection of tests
 - Still could be bugs

Continuous Integration: the Journey

Remotely Deploy

Remotely Test

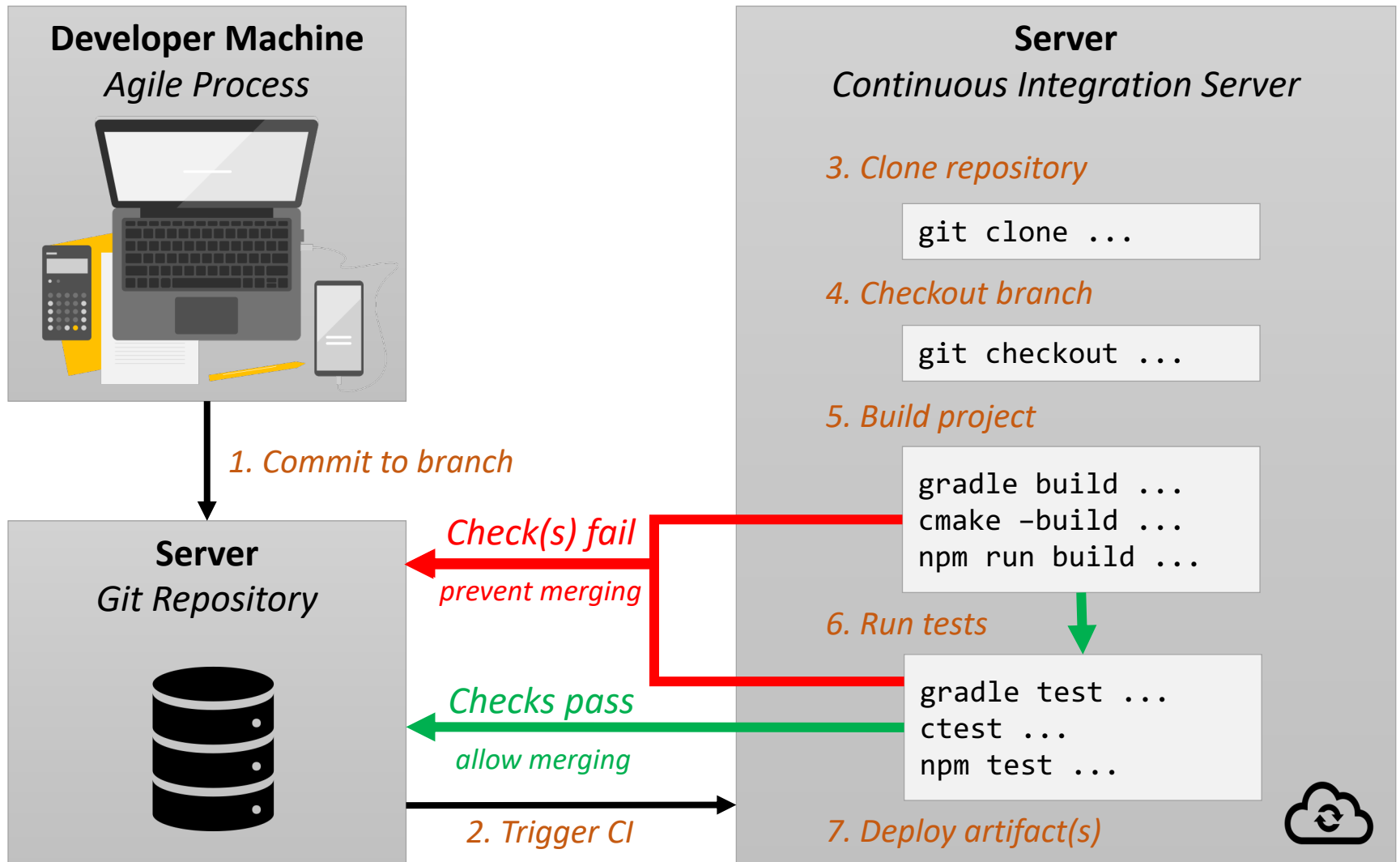
Remotely Build

Locally Test

Remotely Test

- Build and run tests on their own device
- When code is committed to central repository, automatically build the project.
 - **If errors are reported, then the code cannot be merged**
 - If no errors are reported, then automatically run all the tests
 - **If errors are reported, then the code cannot be merged**
- Review other developers' pull requests
 - Still could be bugs

Continuous Integration



Steps to Success

1. Maintain a Code Repository

- Github, Gitlab, Bitbucket, etc. are all services which have a code repository and project management features
- Developers commit the repository on feature branches
- Feature branches are reviews and merged into the working project
- Every **N** weeks, a new version of the project is released

Steps to Success

2. Automate the Build

- Manually compiling or relying on an IDE to build for you (e.g. Eclipse) is not acceptable
- Adopt a build system/framework
e.g. maven, gradle, cmake, npm, pybuilder
- Enables a single command to be invoked for project build. Fewer steps -> Lower Likelihood of Errors

Steps to Success

3. Make the Code Self-Testing

- This requires the creation of tests
- Adopt a testing framework
e.g. JUnit, GTest/GMock, Mocha, pytest

- Run a single command to test everything

```
gradle test    # Java + JUnit w/ Gradle
ctest         # C/C++/FORTRAN w/ CMake
pytest        # Python
```

Steps to Success

4. *Create a Test Environment*

Reproducible server/configuration that can:

- Download your repository
- Build the project
- Run all the project's tests

Often done through the usage/creation of Containers

Containers are like an operating system + software stack...

- Runs on “the cloud” (someone else's computer)
- Contains only the software that you want
- Is completely reproducible

Steps to Success

5. *Keep the Builds Fast*



- Minimize Latency
 - Lowest end-to-end time
- Maximize Throughput
 - Multiple builds concurrently(?)

Steps to Success

6. *Trigger Builds from Commits*

- Often referred to as hooks or actions
- Code repositories can interact with **CI Services**
- **Examples:** Travis CI, Circle CI, Github Actions

Steps to Success

7. *Force Checks to Pass before Merging*

- Add a step to your workflow which requires your code to build and have all checks passing prior to merging
- ***Checks to Include:***
 - Project Builds
 - Project Tests Pass
 - Project Code Coverage doesn't Decrease
 - Style/Linting checks

Continuous Integration Services

Travis CI

`.travis.yml` config file

```
language: java
jdk: oraclejdk11
after_success:
  - mvn clean test jacoco:report coveralls:report
```

Example Output:

✓ **feature/dan/code-coverage** little more code coverage

🕒 #371 passed

🔄 Restart build

🕒 Commit 818aee1 [↗](#)

🕒 Ran for 1 min 11 sec

🔗 Compare 8a82eeb..818aee1 [↗](#)

📅 10 months ago

🔗 Branch feature/dan/code-coverage [↗](#)



🔗 </> JDK: oraclejdk11 Java

🔗 AMD64

Continuous Integration Services

GitHub Actions

`.github/workflows/*.yml` config file(s)

Example Output:

```
name: Unit Tests

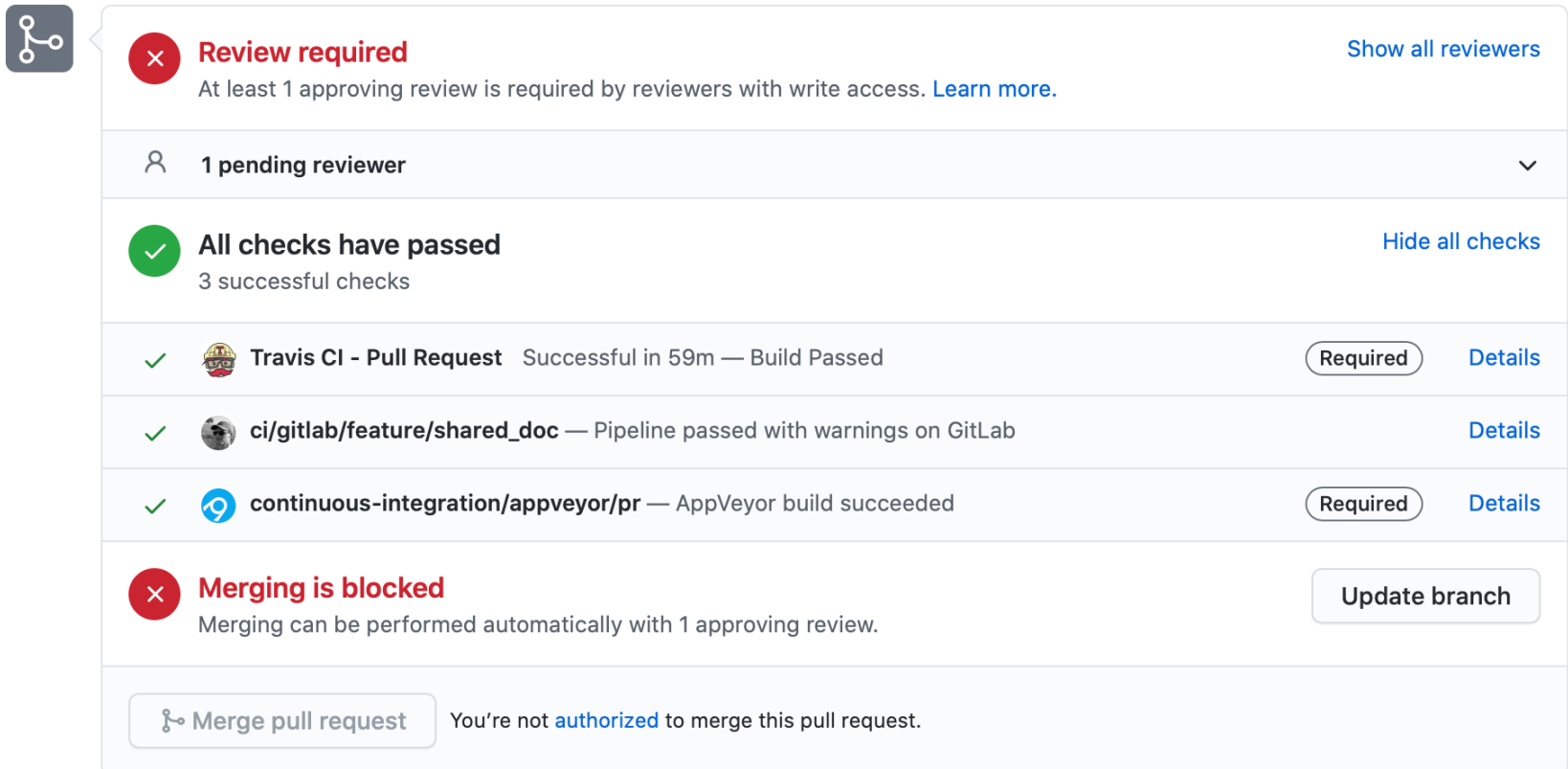
on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up JDK 1.8
        uses: actions/setup-java@v1
        with:
          java-version: 1.8
      - name: Build with Maven
        run: mvn -B package --file pom.xml
      - name: Code Coverage
        uses: codecov/codecov-action@v1.0.6
        with:
          token: ${{ secrets.CODECOV_TOKEN }}
          fail_ci_if_error: true
```


Event ▾ Status ▾ Branch ▾ Actor ▾



✓ Update UserGuide.md Java CI with Maven #159: Commit 30b9933 pushed by bjnulli1	develop	📅 5 months ago 🕒 1m 7s	...
✓ Update UserGuide.md Unit Tests #47: Commit 30b9933 pushed by bjnulli1	develop	📅 5 months ago 🕒 4m 6s	...
✓ Add files via upload Java CI with Maven #158: Commit 5f8968c pushed by bjnulli1	develop	📅 5 months ago 🕒 2m 54s	...


Augmenting Pull Request Checks







A screenshot of a pull request status summary. It features a dark gray icon of a branching diagram on the left. The main content is a light gray box with rounded corners, divided into several sections. The top section has a red circle with a white 'x' and the text 'Review required' in red, followed by a message and a 'Show all reviewers' link. Below this is a section with a person icon and '1 pending reviewer'. The next section has a green circle with a white checkmark and 'All checks have passed' in bold, followed by '3 successful checks' and a 'Hide all checks' link. This is followed by three rows of check status: 'Travis CI - Pull Request' (Successful in 59m — Build Passed) with a 'Required' badge and 'Details' link; 'ci/gitlab/feature/shared_doc' (Pipeline passed with warnings on GitLab) with a 'Details' link; and 'continuous-integration/appveyor/pr' (AppVeyor build succeeded) with a 'Required' badge and 'Details' link. The bottom section has a red circle with a white 'x' and 'Merging is blocked' in red, followed by a message and an 'Update branch' button. At the very bottom is a 'Merge pull request' button with a branching diagram icon and a message stating 'You're not authorized to merge this pull request.'



 **Review required** [Show all reviewers](#)
At least 1 approving review is required by reviewers with write access. [Learn more.](#)


 **1 pending reviewer** 


 **All checks have passed** [Hide all checks](#)
3 successful checks

  **Travis CI - Pull Request** Successful in 59m — Build Passed Required [Details](#)

  **ci/gitlab/feature/shared_doc** — Pipeline passed with warnings on GitLab [Details](#)

  **continuous-integration/appveyor/pr** — AppVeyor build succeeded Required [Details](#)

 **Merging is blocked** Update branch
Merging can be performed automatically with 1 approving review.

 **Merge pull request** You're not [authorized](#) to merge this pull request.

Required for Sprint 3

- Continuous Integration for your Project
 - Can use GitHub Actions or Travis CI
 - Must **Build Project**
 - Must **Run Project** tests
 - Schedule a meeting with *Project Manager* to change requirements for merge workflow with Pull Requests
- Code Coverage Reporting
 - Talk more about this next week
 - Recommended (Required for Sprint 4)
 - Helps identify what code is being run/tested