Calendar

Week 13, Mon. April 9 through Fri April 13

- Lecture on Static/Dynamic Code Analysis
- Lecture on Continuous Integration
- Lab 11 this week on Automated Unit Testing
- Guest Lecture on Friday, April 13 – Software and the Law
- Homework 4 (Web Services) in progress, due April 15 at midnight
- Milestone 5 (Project test planning) assigned

Calendar

Week 14, Mon April 16 through Fri April 20

- Guest Lecture on Monday, April 16 – Professional Software Testing, working at Google
- Lab 12 this week on Continuous Integration
- Lecture on Documenting your Code
- Lecture on Presenting your Project
- Midterm Review and course wrap-up
- Milestone 5 due Sun April 22 at midnight

Calendar

Week 15, Mon April 23 through Fri April 27

- No lab this week
- 2nd MidTerm on Monday Apr 23
- Lecture on Documenting your Code
- Lecture on Presenting your Project, Midterm Review
- Milestone 5 due April 22
- Presentations (Milestone 6) during Lecture time, Fri April 27

Calendar

Week 16, April 30 – May 3

- Presentations (Milestone 6) during Lecture time, Mon April 30
- Presentations (Milestone 6) during recitation sections Monday through Thursday.
- No classes Friday, May 4 (“Reading Day”)

Week 17, May 7-8-9 (Finals Week)

- Milestones 7 & 8 due by Wednesday, May 9, midnight

Milestone 5 – Test Plan

Milestone 5 due Sunday, April 22 midnight (11:59 p.m.)

The test plan is the fifth deliverable required from the team. All the information listed below must be included and the test files should be in place by midnight on the due date.

User Acceptance Test Plans	Create a document that describes how at least three features within your finished product will be tested. The test plan should include specific test cases (user acceptance test cases) that describe the data and the user activity that will be executed in order to verify proper functionality of the feature.
Automated Test Cases	Provide a link to the tool you use to automate testing, or explain how to run the automated test cases or schedule time with the TAs to demonstrate your automated tests. Provide a copy of the output showing the results of the automated test cases running.

Submission format: This project milestone 5 submission should be a PDF named ProjectMilestone5_<TeamName> included in your github repository containing the following components:

- Who: List of people on the team
- Title: of the project
- Automated Tests: Explanation and Screenshot (see above)
- User Acceptance Tests: Copy of at least three UAT plans

Milestones 6-7-8

Milestone 6 due during Week 15-16

(Fri April 27 – Thu May 3)

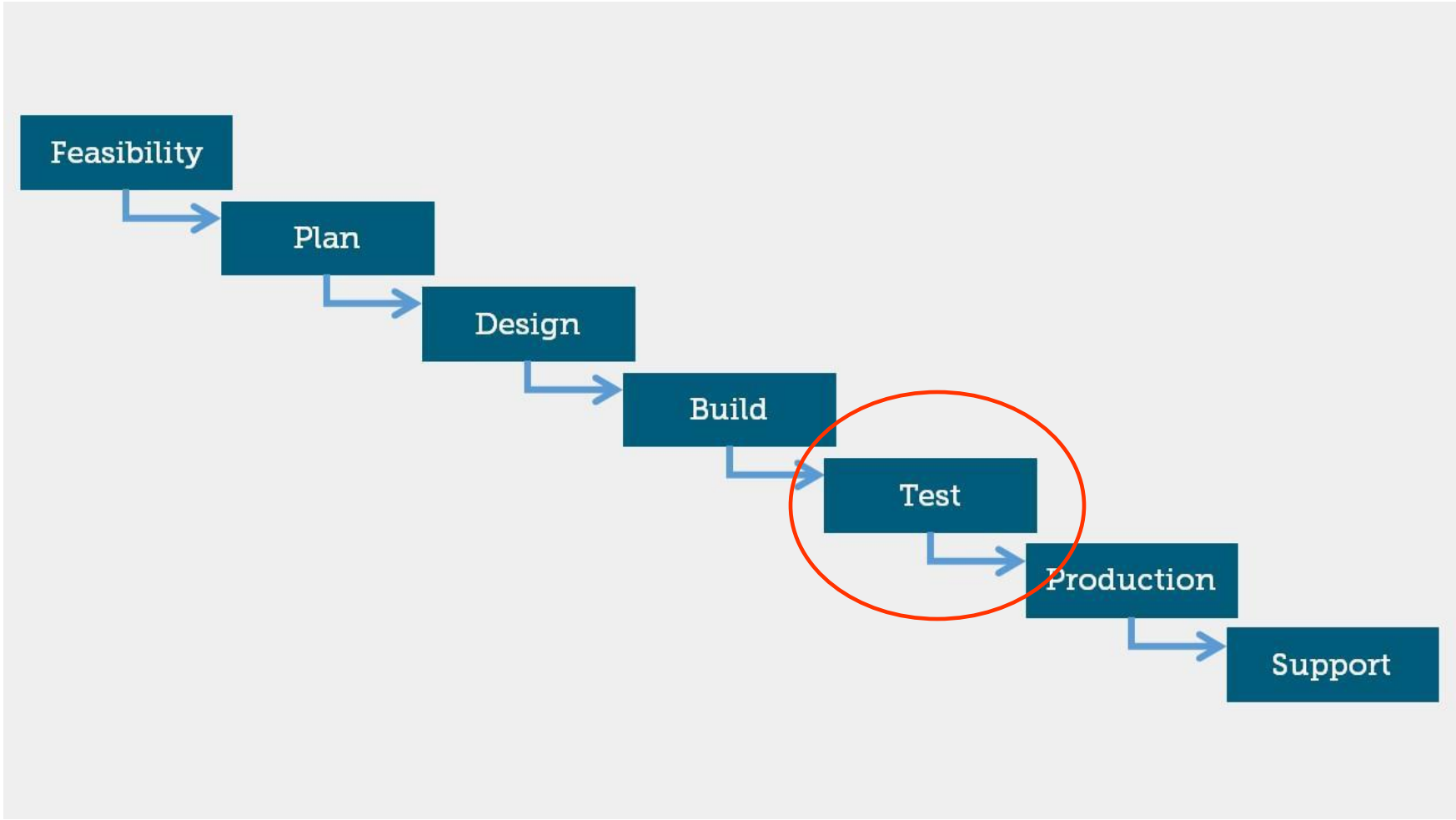
- Ten Minute Team Presentation
- During your lab time, OR, during lecture on Friday April 27 or Monday April 30
- Sign Up for Time Slot on Moodle
- Extra Credit (+5) for Presenting During Lecture Slots (up to 5 team presentations)
- All Members Must Be Present
- All Members Must Present

Milestones 6-7-8

Milestone 7: Final Project Submission (TEAM) due during week 17 by May 9

Milestone 8: Peer Evaluation & Project Reflection (individual) due during week 17 by May 9

Following the Waterfall Methodology



Continuous Integration

Traditional "Waterfall" Testing Practices

- Testing occurs as a project phase towards the end of project
- Lots of lead time for test planning, test case generation, test environment and infrastructure setup
- Test cases don't change often (tied to requirements)
- Cost of creating test cases is borne once
- Few changes to system once it is specified and designed
- Tests are executed periodically
- Initial round of testing to ensure system meet requirements
- Regression testing after any significant change to ensure nothing was broken

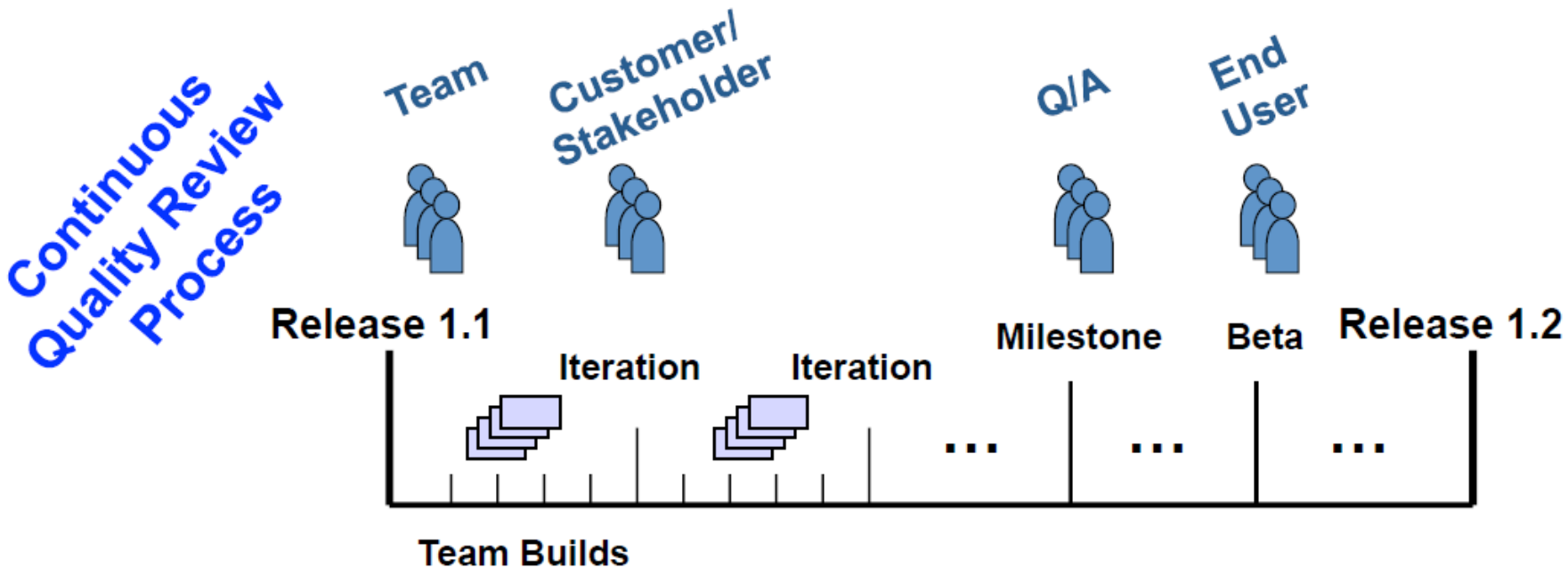
Continuous Integration

Following AGILE Practices

- Development is continuous
- No separate phase for testing
- Develop and test continuously, feature by feature
- Features change, new features come up
- Testing must adapt to changes

Continuous Integration

- Development is continuous



Continuous Integration

"Continuous Integration"

- Follows the Agile Approach
- Definition:
  - Integrate & build the system several times a day
  - Integrate every time a task is completed
- Lets you know every day the status of the full system
- Continuous integration and relentless testing go hand-in-hand.
- By keeping the system integrated at all times, you increase the chance of catching defects early and improving the quality and timeliness of your product.
- Continuous integration helps everyone see what is going on in the system at all times.

Continuous Integration

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily, leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible."

Martin Fowler

https://en.wikipedia.org/wiki/Martin_Fowler

*If **testing** is good, why not do it all the time? (continuous testing)*
*If **integration** is good, why not do it several times a day? (continuous integration)*
*If **customer involvement** is good, why not show the business value and quality we are creating as we create it (continuous reporting)*

Why do Continuous Integration?

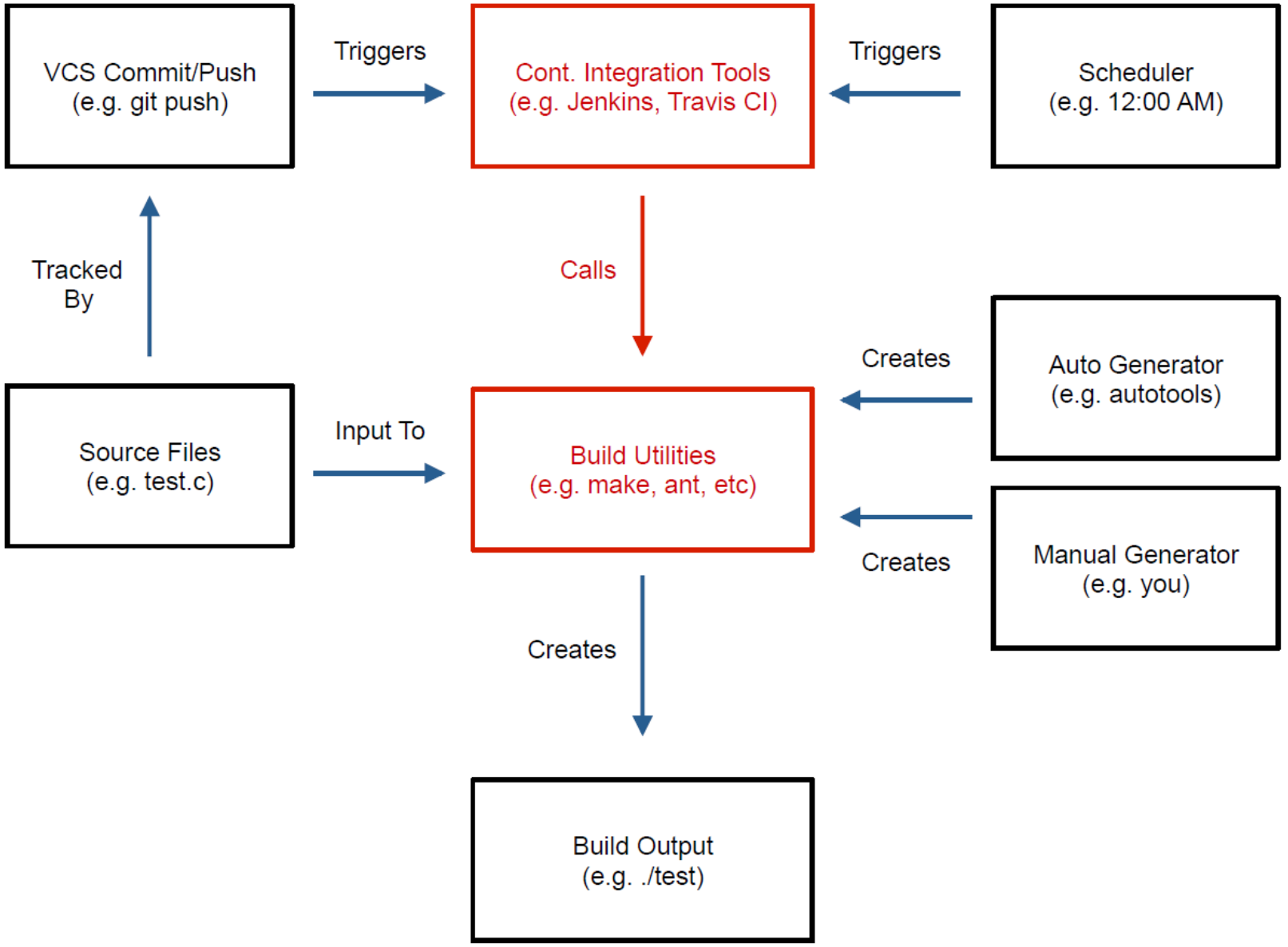
- Speed Up Each Feature/Sprint
- Automate Deployment
- Guarantee that Necessary Tests Are Being Run
- Track Build and Test Status
- Immediate Bug Detection
- Yields more Bug-Free Code
- A “Deployable” system at any given point
- Record of the history/evolution of the project

Why do Continuous Integration?

At regular intervals (ideally at every commit), the system is:

- Integrated
- All changes up until that point are combined into the project Build
- The code is compiled into an executable or package
- All code is Tested (regression)
- Automated test suites are run
- All code is Archived
- All code is versioned and stored so it can be distributed as is, if desired
- All code is Deployed
- All code is Loaded to an environment where the developers can interact with it

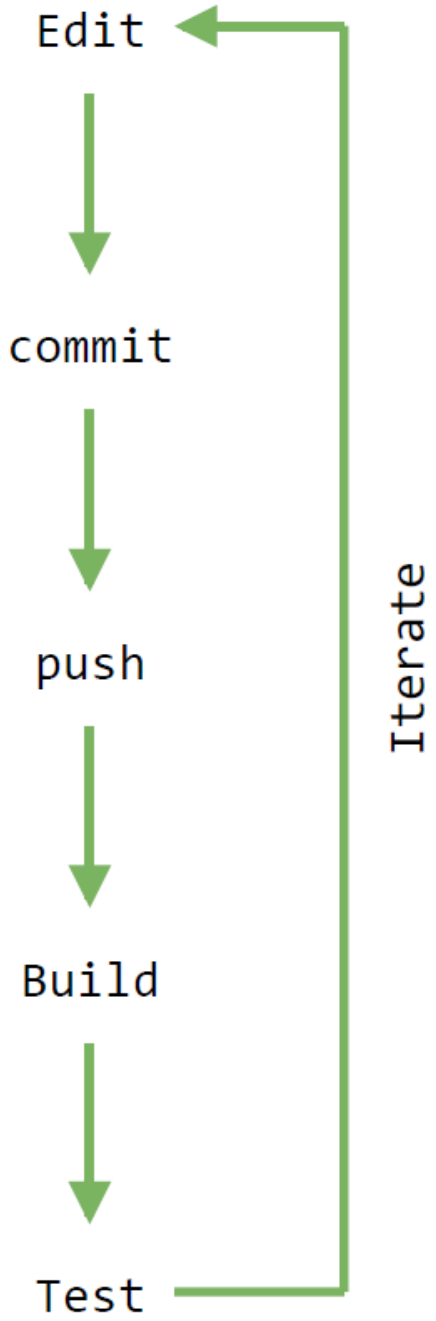
Continuous Integration



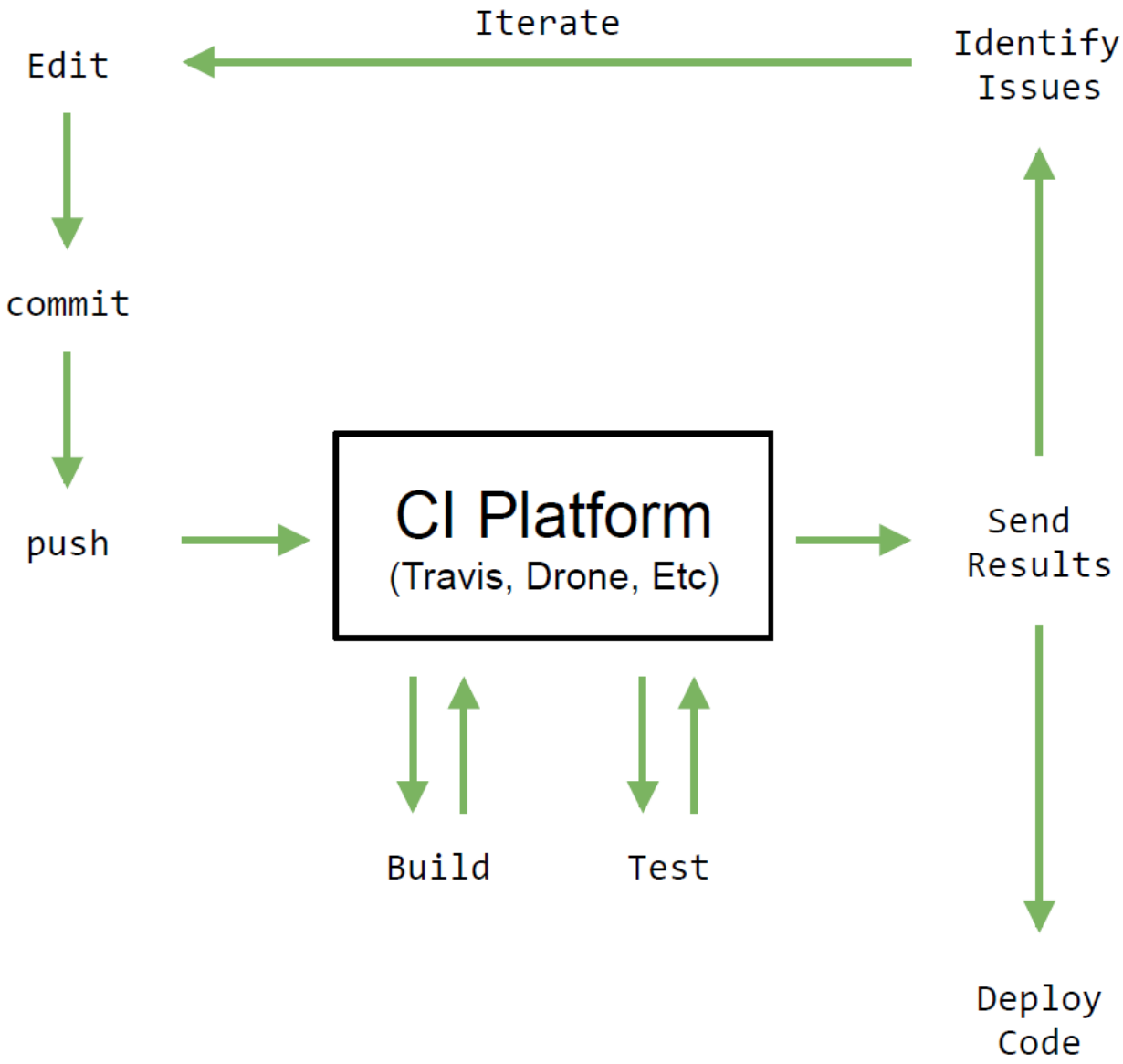
Fowler's 10 Best Practices for CI

1. Maintain a Single Source Repository
2. Automate the Build
3. Make your Build Self-testing
4. Everyone Commits Every day
5. Every Commit should Build the Mainline on an Integration Machine
6. Keep the Build Fast
7. Test in a Clone of the Production Environment
8. Make it easy for Anyone to get the Latest Executable
9. Everyone can see what's Happening
10. Automate Deployment

Continuous Integration



Continuous Integration



Continuous Integration

drone.io: https://drone.io/

Public Projects: Free

Private Projects: Paid

Concurrent Builds: 1

VCS: GitHub, Bitbucket, Google Code

Travis CI: https://travis-ci.org/

Public Projects: Free

Private Projects: Paid

Concurrent Builds: A Few

VCS: GitHub