CS 4530: Fundamentals of Software Engineering Lesson 5.3 Continuous Integration

Rob Simmons Khoury College of Computer Sciences

© 2025 Released under the <u>CC BY-SA</u> license

Continuous Integration (CI) provides global feedback on local changes

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



A CI process is a software pipeline



Other developers' changed code

CI may be triggered by commits, pull requests, or other actions



Automating Feedback Loops is Powerful

Consider tasks that are done by *dozens* of developers (e.g. testing/deployment)

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE? (ACROSS FIVE YEARS)

| | | HOW OFTEN YOU DO THE TASK | | | | | | |
|---------------------|--------------------|---------------------------|---------------|---------------|---------------|---------------|--|--|
| | 50/ _{DAY} | 5/DAY | DAILY | WEEKLY | MONTHLY | YEARLY | | |
| 15 | ECOND 1 DAY | 2 HOURS | 30 MINUTES | 4 MINUTES | 1 MINUTE | 5 SECONDS | | |
| 5 SE | CONDS 5 DAYS | 12 HOURS | 2 HOURS | 21 MINUTES | 5 MINUTES | 25 SECONDS | | |
| 30 55 | CONDS 4 WEEKS | 3 DAYS | 12 HOURS | 2 HOURS | 30 MINUTES | 2 MINUTES | | |
| HOW 1 M MUCH 1 M | INUTE 8 WEEKS | 6 DAYS | 1 DAY | 4 HOURS | 1 HOUR | 5 MINUTES | | |
| TIME 5 MI | NUTES 9 MONTHS | 4 WEEKS | 6 DAYS | 21 HOURS | 5 HOURS | 25 MINUTES | | |
| OFF 30 MI | NUTES | 6 MONTHS | 5 WEEKS | 5 DAYS | 1 DAY | 2 HOURS | | |
| 1 | HOUR | 10 months | 2 MONTHS | 10 DAYS | 2 DAYS | 5 HOURS | | |
| 61 | HOURS | | | 2 MONTHS | 2 WEEKS | 1 DAY | | |
| 1 | Day | | | | 8 WEEKS | 5 DAYS | | |

© Randal Munroe/xkcd, licensed CC-BY-SA https://xkcd.com/1205/

Typical CI pipeline

- Set up testing environment
- Set up tests
- Set up multiple input
- Run all tests against all inputs
 - (preferably in parallel)
- Record results and performance in central db



You could set up multiple CI processes

- Run a short test daily
 - or oftener
 - maybe on every commit?
- More comprehensive test less often
 - provides more accurate performance data
- Either way, you know that your integration is working!

Continuous Integration is Highly Configurable

- Determining how to apply CI can be non-trivial for a larger project, all with a cost vs quality tradeoff: what is the cost of automation vs the value of developer time?
- Do we integrate changes immediately, or do a pre-commit test?
- Which tests do we run when we integrate?
- When do we integrate code review?
- How do we compose the system under test at each point?



CI pipelines can automate performance testing



CI pipelines can automate benchmarking

closure

Branch Probes Over Time





Attributes of effective CI processes

- Policies:
 - Do not allow builds to remain broken for a long time
 - CI should run for every change
 - CI should not completely replace pre-commit testing
- Infrastructure:
 - CI should be fast, providing feedback within minutes or hours
 - CI should be repeatable (deterministic)

| All checks have passed 9 successful checks | |
|--|---------|
| Successfu | Details |
| ✓ | Details |
| \checkmark (Build and Test the Grader / test (reference) (push) | Details |
| Succes | Details |
| Ruild and Test the Grader / test (ts-ignore) (nush) | Details |
| Tools: extract_features.py: correct define name for AP_RPM_ENAB | BLED |

Peterbarker committed 5 days ago ×
AP_HAL_ChibiOS: disable DMA on I2C on bdshot boards to free up DMA ch...

 $\square 2$

💮 andyp1per authored and tridge committed 6 days ago 🗙

SITL: Fixed rounding lat/Ing issue when running JSBSim SITL ShivKhanna authored and tridge committed 6 days ago ×

AP_HAL_ChibiOS: define skyviper short board names yuri-rage authored and tridge committed 6 days ago ×

Effective CI processes are run often enough to reduce debugging effort

- Failed CI runs indicate a bug was introduced, and caught in that run
- More changes per-Cl run require more manual debugging effort to assign blame
- A single change per-CI run pinpoints the culprit

| Current Branches Build | History Pull Requests | | More options |
|-------------------------------|--|---|---|
| ✓ master | This patch bumps Alluxio dependency to 2.3.0-2 | -∾- #52300 passed -∞- 36392a2 🖄 | () 10 hrs 49 min 31 sec 2 days ago |
| 1 master | Handle query level timeouts in Presto on Spark | #52287 errored aa55ea7 @ | ① 11 hrs 6 min 44 sec⑦ 2 days ago |
| ! master | Fix flaky test for TestTempStorageSingleStream S_β | #52284 errored 193a4cd @ | ① 11 hrs 50 min 37 sec 2 days ago |
| ✓ master | Check requirements under try-catch | >- #52283 passed >- fff331f ⊘ | ① 11 hrs 3 min 20 sec ② 2 days ago |
| √ master ⊛ Maria Basmanov | Update TestHiveExternalWorkersQueries to create | -∽- #52282 passed -∽- 746d7b5 2 | ① 10 hrs 55 min 37 sec② 2 days ago |
| ✓ master 🥮 Maria Basmanova | Introduce large dictionary mode in SliceDictionar a | -∾ #52277 passed -⊳ a90d97a & | ① 10 hrs 43 min 30 sec② 2 days ago |
| ! master Maria Basmanova | Add Top N queries to TestHiveExternalWorkersQ \ensuremath{u} a | #52271 errored 8b62d43 🖉 | ① 10 hrs 46 min 36 sec☑ 3 days ago |
| X master | Fix client-info test-name output | -∞ #52266 failed -∞ 467277a Ø | () 10 hrs 35 min 49 sec |
| ✓ master © Andrii Rosa | Add Thrift transport support for TaskStatus | -∘- #52263 passed -∘- fc94719 ⊘ | () 11 hrs 13 min 42 sec 77 3 days ago |
| | | | |

🔲 prestodb / presto 💭 🗖

Effective CI processes allocate enough resources to mitigate flaky tests

- Flaky tests might be dependent on timing (failing due to timeouts)
- Running tests without enough CPU/RAM can result in increased flaky failure rates and unreliable builds



Challenges and Solutions for Repeatable Builds

- Which commands to run to produce an executable? (build systems)
- How to link third-party libraries? (dependency managers)
- How to specify system-level software requirements? (containers)
- How to specify infrastructure requirements? (Infrastructure as code)

Build Systems Orchestrate Software Engineering Tasks

- "Orchestrate" -> Execute in the right order, ideally with concurrency, example tasks:
 - Installing dependencies
 - Compiling the code
 - Running static analysis
 - Generating documentation
 - Running tests
 - Creating artifacts for customers
 - Deploying Code
- Example build systems: xMake, ant, maven, gradle, npm...

Dependency Managers Organize External Dependencies

- Addresses this problem: "Before you compile this code, install commons-lang from the Apache website"
- Declare a dependency using coordinates (unique ID of a package plus version)
- Packages are archived in common repositories; fetched/linked by dependency manager
- Dependency managers handle transitive dependencies
- Examples: Maven, NPM, pip, cargo, apt

Specify and Depend on Package Versions with Care

- <u>Semantic Versioning</u> is often expected:
 - Library maintainers expected to indicate breaking changes with version numbers
 - Dependency consumers can specify constraints on versions (e.g. accept 2.0.x)





Continuous Integration Service Models

- Self-hosted/managed on-premises or in cloud
 - Jenkins
- Fully cloud managed
 - GitHub Actions, CircleCI, Travis, many more...
 - Billing model: pay per-build-minute running on SaaS infrastructure
 - "Self-hosted runners" run builds on your own infrastructure, usually "free"