Playbook Methods Repository

# **Continuous Integration / Continuous Delivery (CI/CD)**

Develop software with an automated process that ensures successful integration, repeatable automated building and packaging, and repeatable automated deployment.

### Remote Agility: **•** High

### Linked Tactic(s): Agile Development, Quality Assurance, Release Management, Rolling Release

## Why we do it:

Continuous Integration facilitates rapid software development by reducing integration friction (merges) of work in progress from multiple sources. By automating integration, building, and testing multiple times daily, it is ensured that the codebase remains cohesive and functional at all times throughout its evolution.

Continuous Deployment facilitates rapid release to production, accelerating the customer feedback loop and iterative development process. By relying on a well-tested code base, manual gating and releasing can be replaced by a fully automated pipeline only halted by test failures.

Through reliance on automation the combined approach of CI/CD practices reduce the risk of deploying an application by exchanging unwieldy agglomerations of work for small, predictable increments.

## 

## When to apply it:

* [New Project Codebases](https://sites.google.com/connectedlab.com/wiki/guilds-departments/engineering/checklists/checklist-new-project-codebases#h.p_6XqcXReFeVXa): Thoughtworks recommends applying this method, along with linked tactics, to any new codebase.
* Complex Integrations: While it’s easiest to apply these practices with new code, they are also great tools in simplifying integration and deployment processes on existing, complex projects. Continuous Integration can be a solution to protracted and challenging merge phases of a project.
* Risky Deployments: Continuous Deployment can increase predictability and reduce risk of bringing existing projects to production.
* Greater Agility: CI/CD reduces the length of product iterations and increases the frequency of feedback by reducing time between deployments, facilitating higher agility by providing more opportunity for course correction.

## Best Practices & Considerations:

CI/CD combines well with pairing and test-driven development. Continuous Integration advocates committing directly to the main branch which fits well with the practice of replacing code reviews (or pull requests) with a pair of engineers, who are constantly peer-reviewing the work of their pair. Both CI and CD rely heavily on automation; a well tested codebase is a requirement and can be facilitated by test-driven development.

Feature Flagging: Feature flags can be a valuable component in a CI/CD pipeline. Flags facilitate CI/CD by decoupling code changes from feature launches - new code, for incomplete features, can be integrated into the code base and even deployed to production without being revealed to the user or impacting the product.

## Responsible roles:

* Software Engineer (SE): the SE writes tests, writes code to pass the tests, and refactors code, merging their changes to the main branch several times daily ensuring that it integrates well with changes from others.
* QA Automation Engineer (QA): the QA will employ the QA Testing Strategy method to help the team determine the approach to testing and automation, enabling the automated CI/CD pipeline. Both QA and SE can work together to build the CI/CD pipeline, including setup and configuration of tooling.

## Tools:

* [Jenkins](https://www.jenkins.io)
* [Fastlane](https://fastlane.tools)
* [CircleCI](https://circleci.com)
* Online tools/platforms/services
  + xx
* Websites
  + xx
* Databases
  + xx
* Other
  + xx

## 

## Thoughtworks Examples - Linked

* Client working docs, airtable, miro/mural boards
  + [Project Saffron Mobile CI/CD Summary](https://docs.google.com/document/d/1VBHtOKFh234hr8_3rpzwoxRnoD95RxifE43mXSaEcI0/edit)
  + VTS
  + Peloton
* Client polished presentations/deliverables
  + xx
* Internal assets - clinic materials / guild docs
  + [New Project Codebases - Before Coding Starts](https://sites.google.com/connectedlab.com/wiki/guilds-departments/engineering/checklists/checklist-new-project-codebases#h.p_6XqcXReFeVXa) (wiki)
  + [Product Thinking Playbook - Sample Plays: Release](https://sites.google.com/connectedlab.com/wiki/how-we-work/product-thinking-playbook/sample-plays#h.p_uZvPc2Xhcww3) (wiki)
  + [Configuring Jenkins](https://sites.google.com/connectedlab.com/wiki/guilds-departments/engineering/platforms/qa/configuring-jenkins) (wiki)

## 

## Learn more: How we do this?

* Templates (docs, decks, sheets, miro, etc.)
  + xx
* How-To Resources (external or internal)
  + xx
* Outside References (articles, books, etc.)
  + [Martin Fowler - Continuous Integration](https://martinfowler.com/articles/continuousIntegration.html)
  + [Martin Fowler - Continuous Delivery](https://martinfowler.com/bliki/ContinuousDelivery.html)
  + [MinimumCD.org](https://minimumcd.org/minimumcd/)
  + [Trunk Based Development](https://trunkbaseddevelopment.com)
* Sub-set Activities
  + xx

## 