Automated Test Framework (ATF)

Art of Testing with ServiceNow

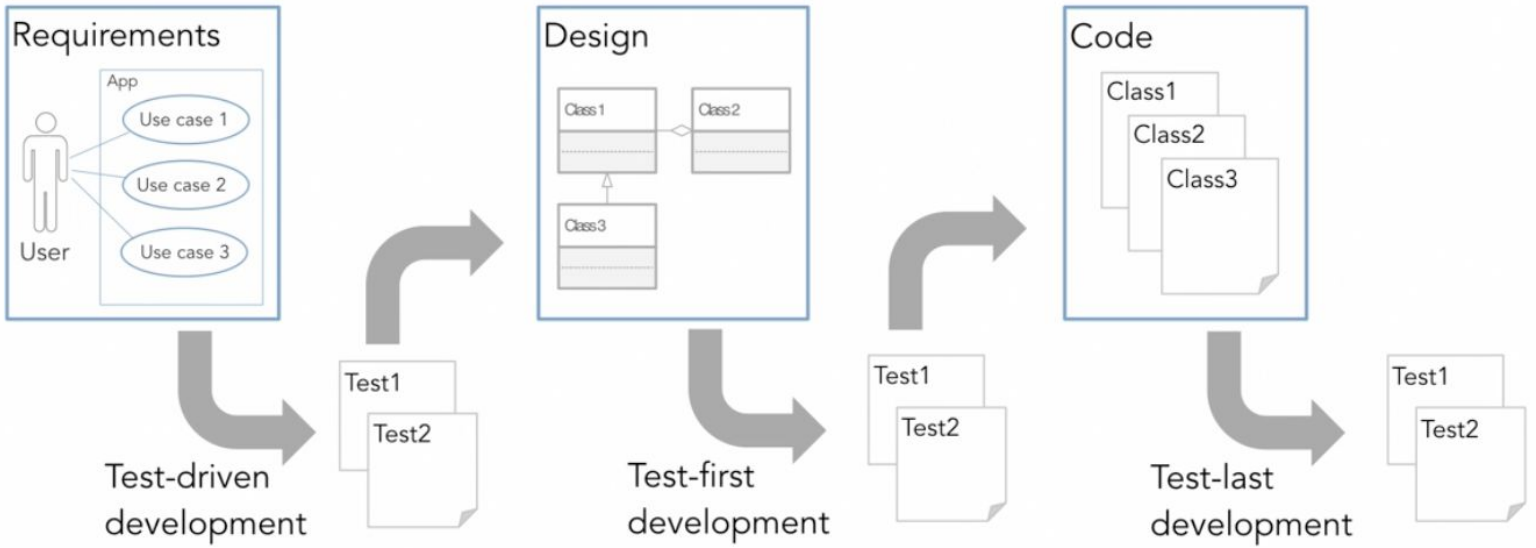
Quality Code Starts with Testability

"Programmers shouldn't have to guess whether software is working correctly. They should be able to prove it, every step of the way."

– Neelam Dwivedi

(Associate Professor at University of Oklahoma)

Quality Code Starts with Testability



Test-Driven Development (TDD)

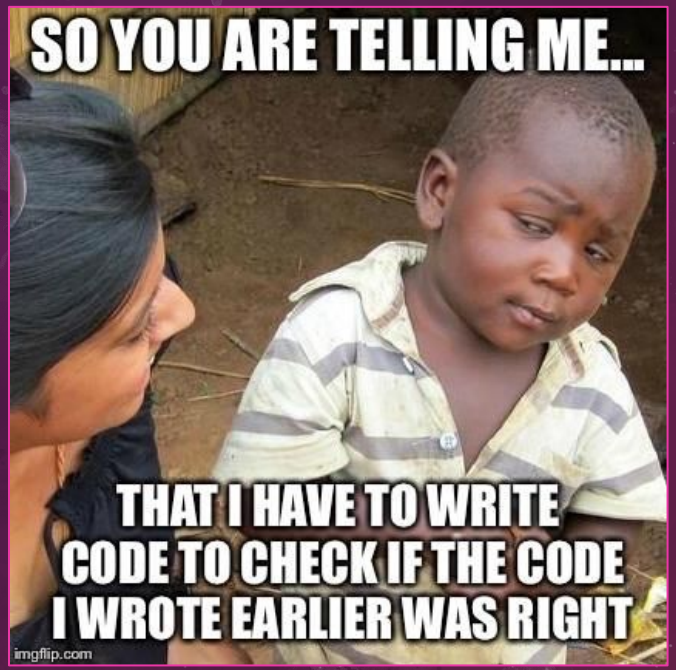
- Test-Driven Development (TDD) is a software development approach where tests are written before the actual code.
- The main goal of TDD is to improve the quality of the software and ensure that the code meets the requirements.

Test-Driven Development (TDD)

More than the act of testing, the act of designing tests is one of the best bug preventers known.

— Boris Beizer —

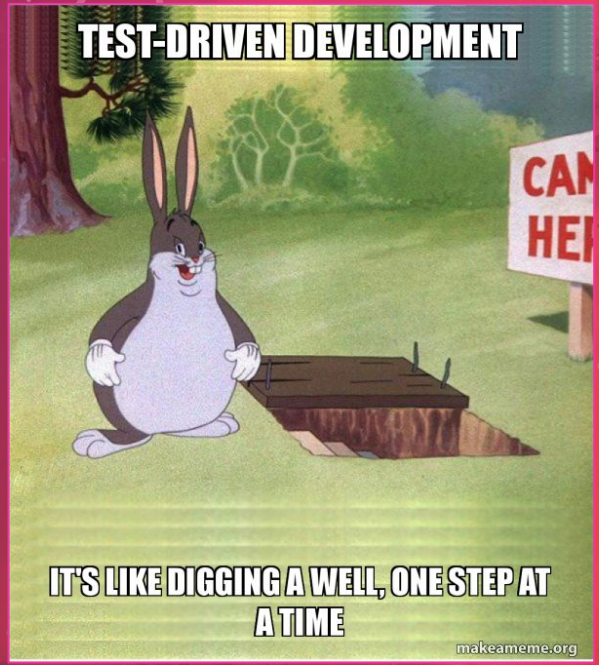
Benefits of TDD



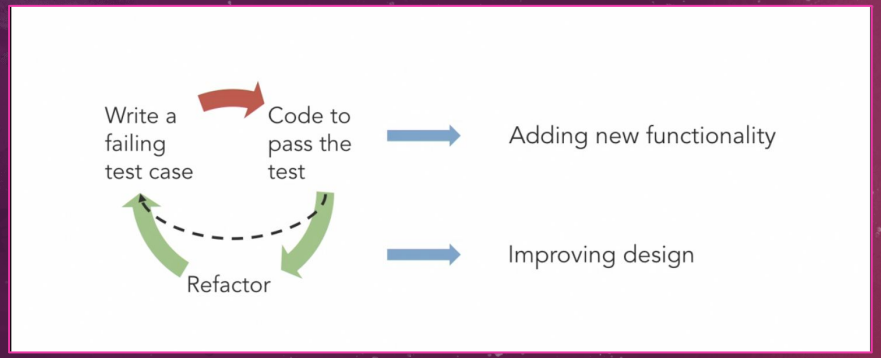
- **Improved Code Quality:** Writing tests first helps clarify requirements and leads to better-designed code.
- **Fewer Bugs:** Since tests are written for every piece of functionality, TDD helps catch bugs early in the development process.
- **Documentation:** Tests serve as documentation for the code, making it easier for new developers to understand the intended behavior.
- **Confidence in Refactoring:** With a comprehensive suite of tests, developers can refactor code with confidence, knowing that any regressions will be caught by the tests.

Test-Code-Refactor Cycle

- "Test-driven development (TDD) is a software development process that relies on the repetitive cycle of writing"
- TDD primarily involves unit testing, which tests the smallest units of code.
- The fundamental cycle in TDD involves writing a failing test case, writing code to pass the test, and then refactoring the code to improve its structure without changing its behavior.

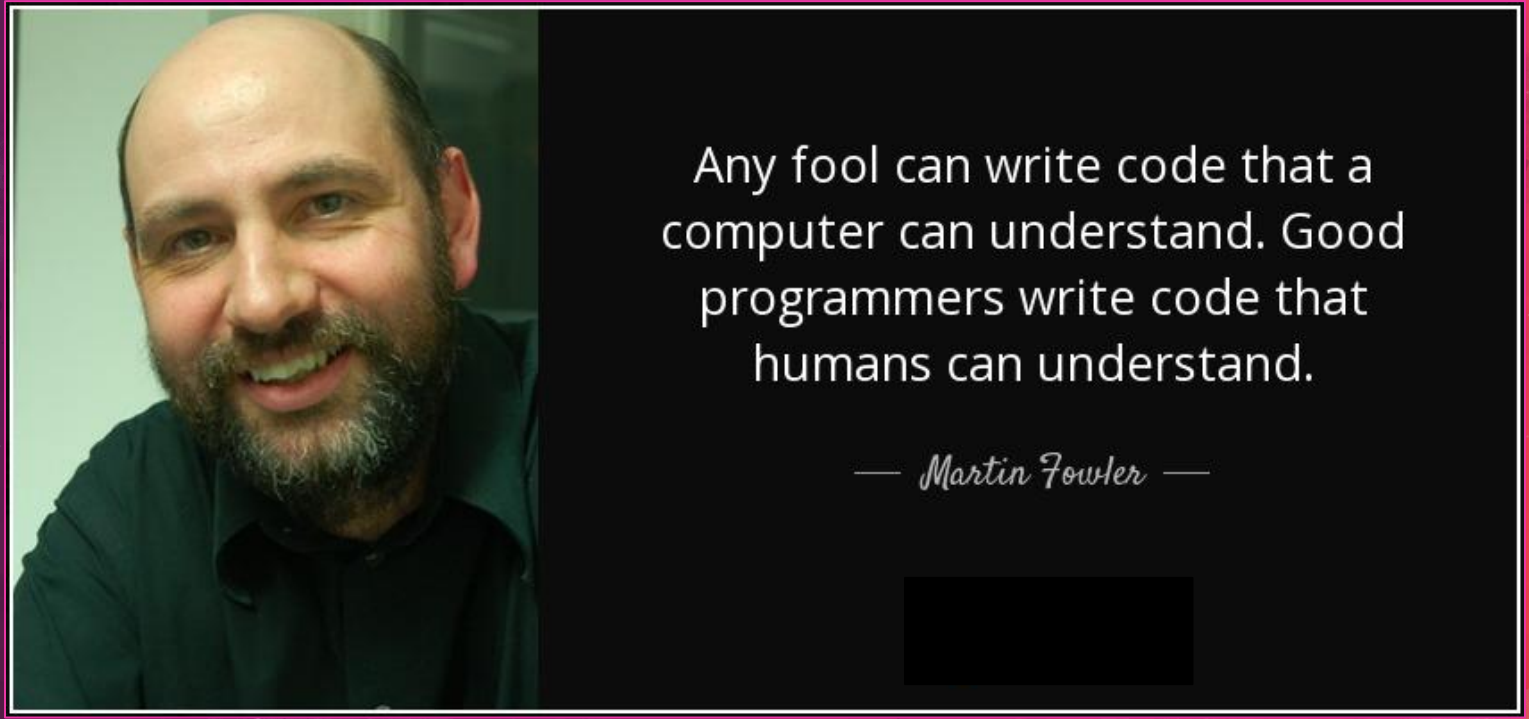


Red-Green-Refactor

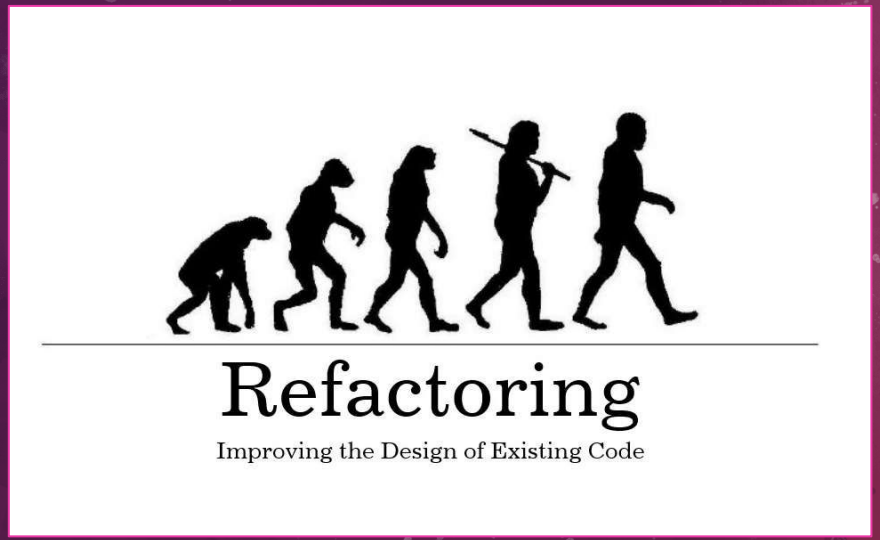


1. **Red:** Write a test for a new feature or functionality. The test should fail because the feature is not yet implemented.
2. **Green:** Write the minimum amount of code necessary to make the test pass. The focus is on getting the test to pass, not on writing perfect code.
3. **Refactor:** Clean up the code while ensuring that all tests still pass.

Refactoring



Refactoring



"Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior."

- This step involves improving the code structure, readability, and performance without changing its behavior.
- Refactoring aims to make code easier to read and cheaper to modify.
- It's done when code becomes redundant, hard to read, or doesn't feel right.

There are no strict rules

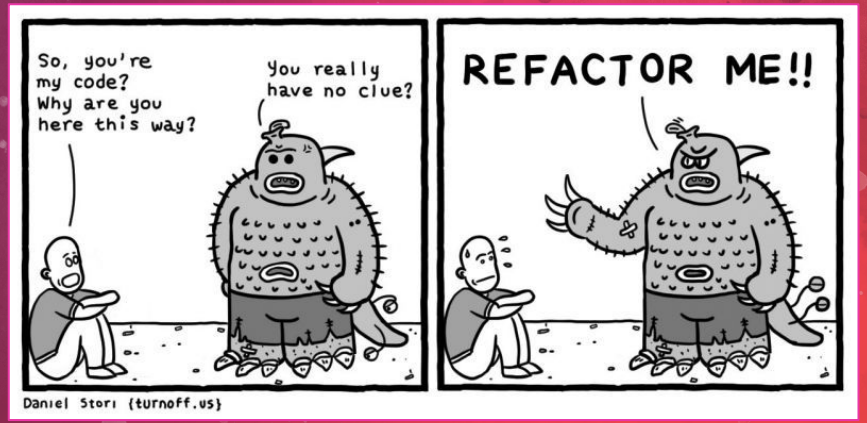
I often notice that the resulting code, while it works, isn't as clear as it could be.

I then refactor it into a better shape so that when I (or someone else) return to this code in a few weeks time, I won't have to spend time puzzling out how this code works.

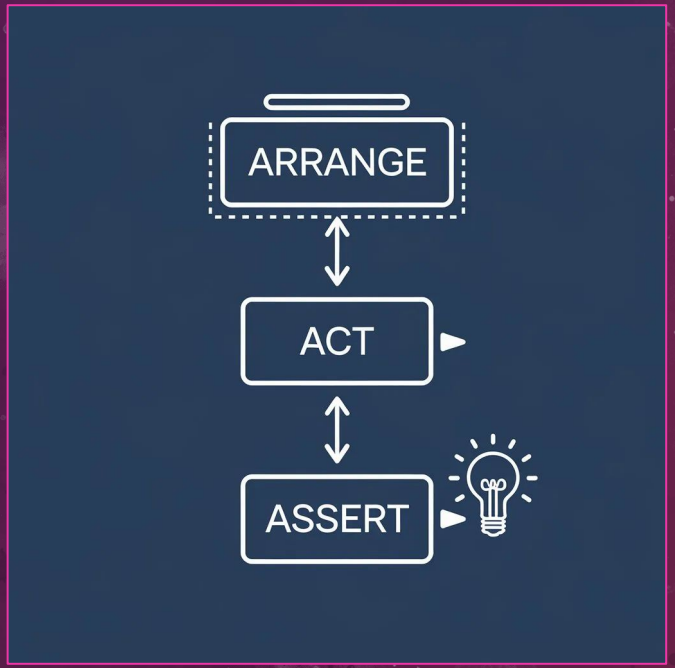
If I struggle to understand this code, I refactor it so I won't have to struggle again next time I look at it.

If there's some functionality buried in there that I need, I refactor so I can easily use it.

- Martin Fowler (Author, Refactoring)



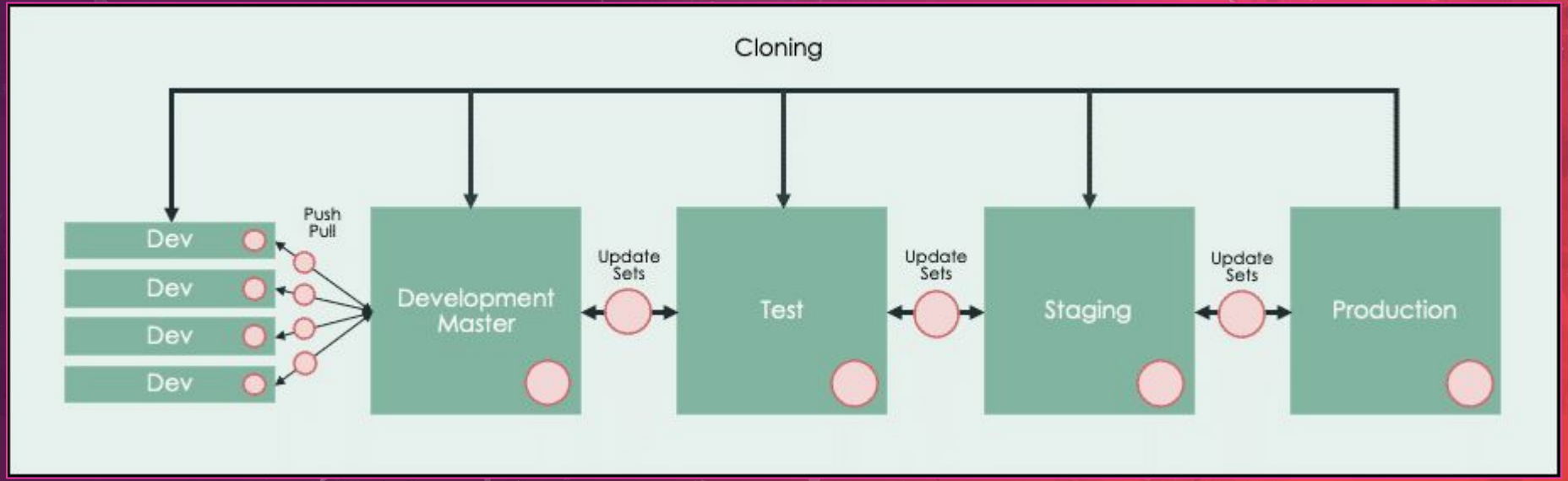
Arrange, Act, Assert Pattern



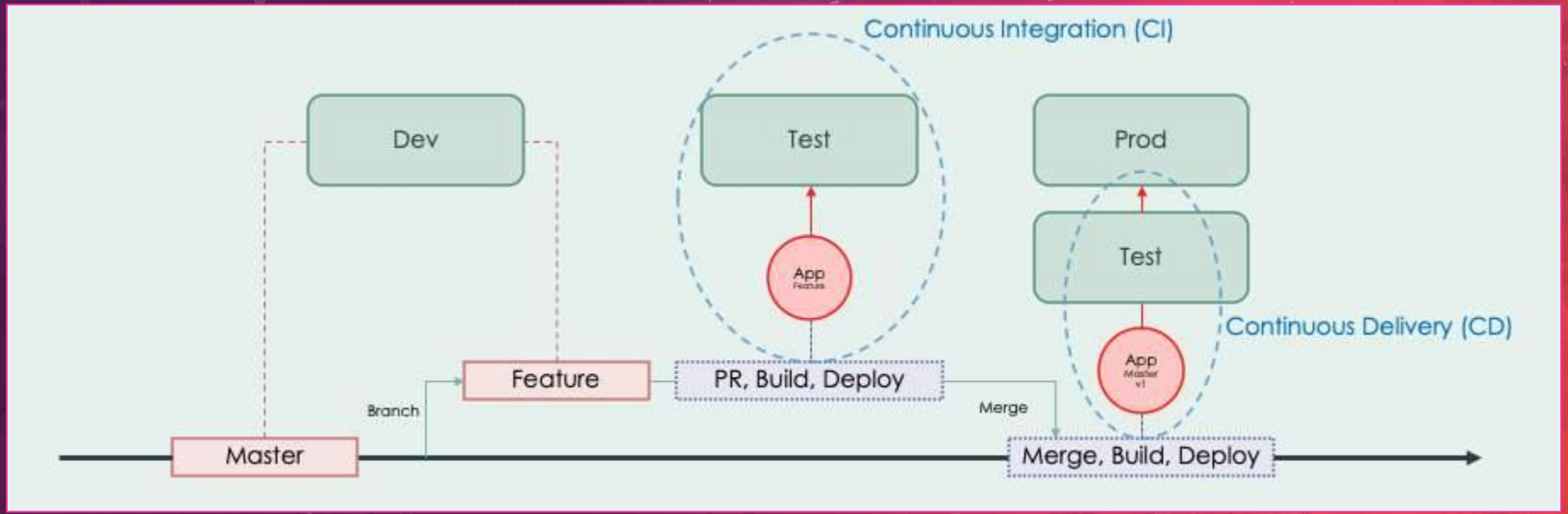
Each test case typically follows this pattern:

1. **Arrange:** Set up the elements required for the test.
2. **Act:** Perform the action you want to test.
3. **Assert:** Verify that the result is as expected.
    a. Assertions are used in programming to ensure that a certain condition is true at a specific point in the program.
    b. If the condition is false, an assertion error is thrown.

Environment Development Model 1

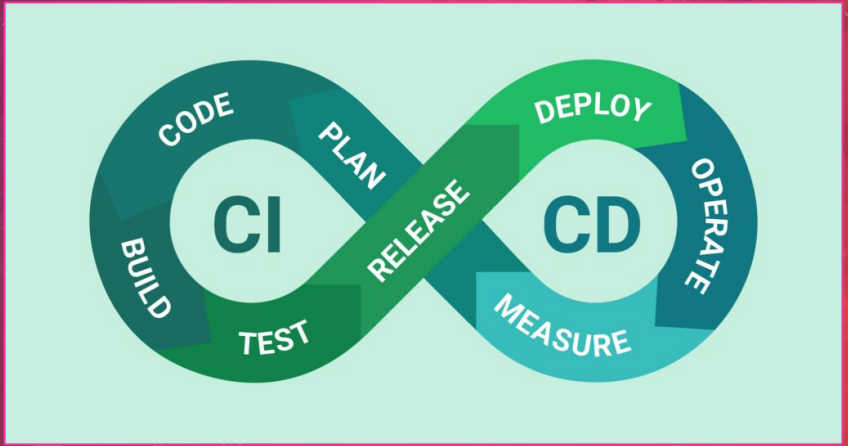


Environment Development Model 2



Continuous Integration and Continuous Delivery (CI/CD)

"Continuous Integration and Continuous Delivery (CI/CD) is a software development practice that aims to improve the quality and speed of software development by integrating and testing code changes frequently."

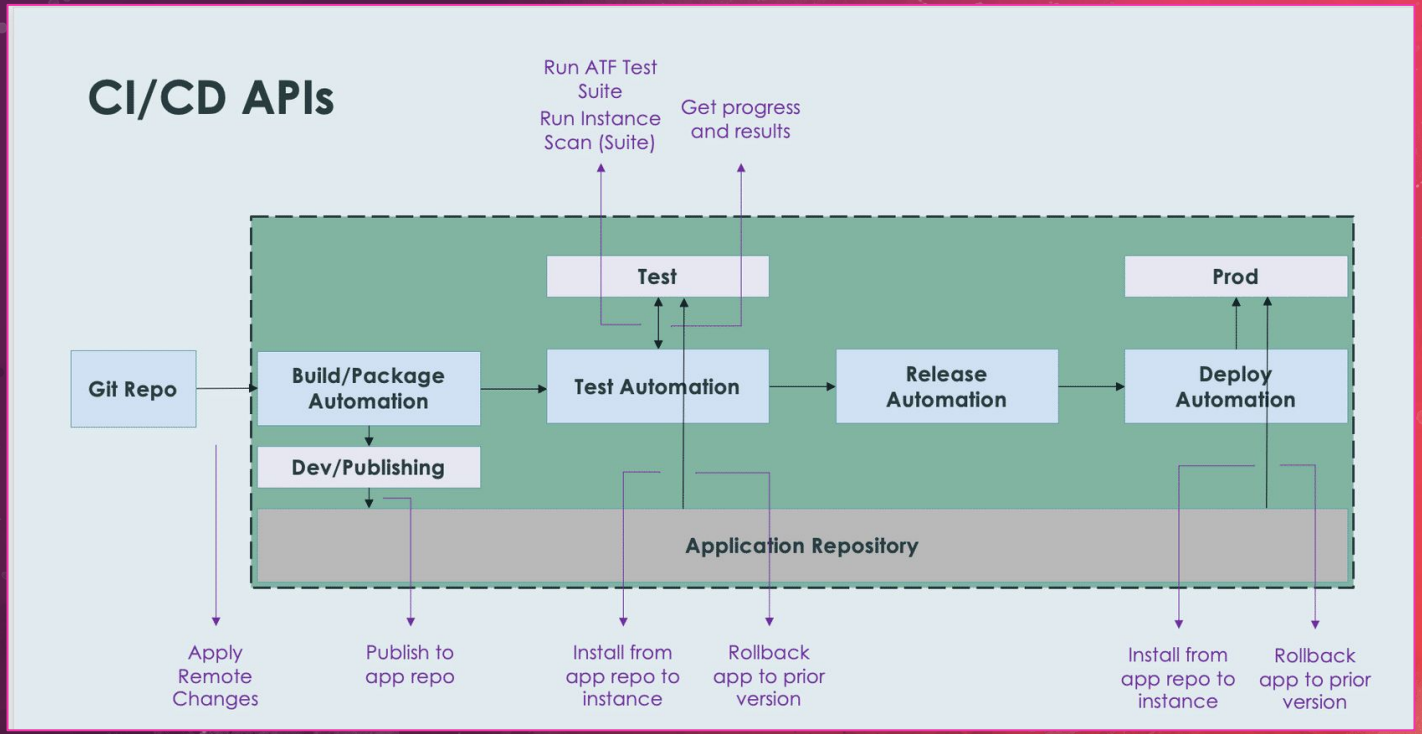


Continuous Integration and Continuous Delivery (CI/CD)

CI and CD stands for Continuous Integration and Continuous Delivery.

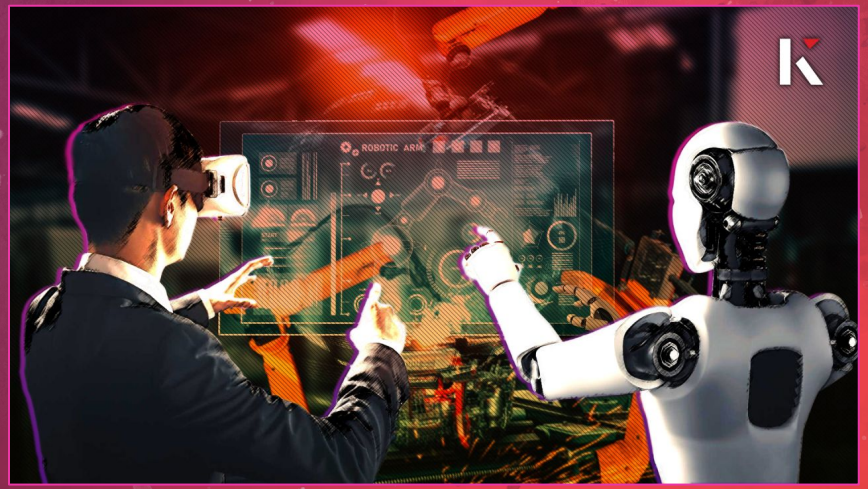
- **Continuous Integration (CI)** : This involves automating the build and test process for code changes, ensuring that the code is integrated and tested frequently.
- **Continuous Delivery (CD)** : This involves automating the deployment of software changes to production environments, ensuring that the software is delivered quickly and reliably.

Continuous Integration and Continuous Delivery (CI/CD)

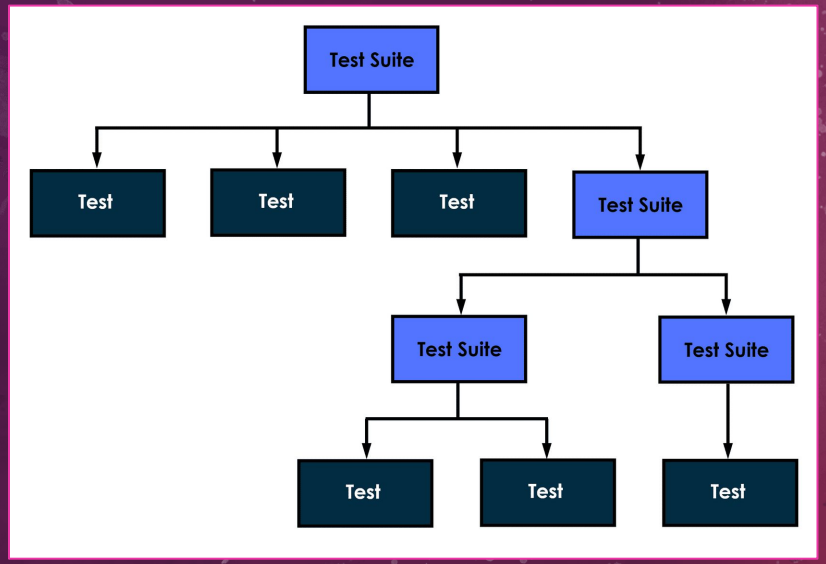


Automated Test Framework (ATF)

- The Automated Test Framework (ATF) is a ServiceNow application to test within a ServiceNow instance.
- You can create and run automated tests to confirm that your instance works after making a change.
- On average, manual testing consumed 43% of time and resources spent on upgrades
- Leveraging the power of ATF, you will be able to reduce the amount of manual testing required to deliver ServiceNow instance changes quickly and effectively.



Building Blocks

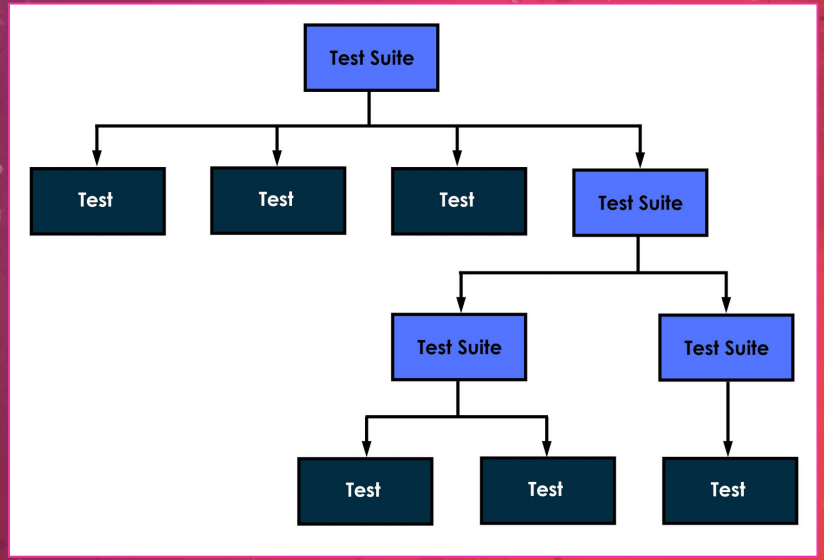


The important elements used to develop ATF testing include:

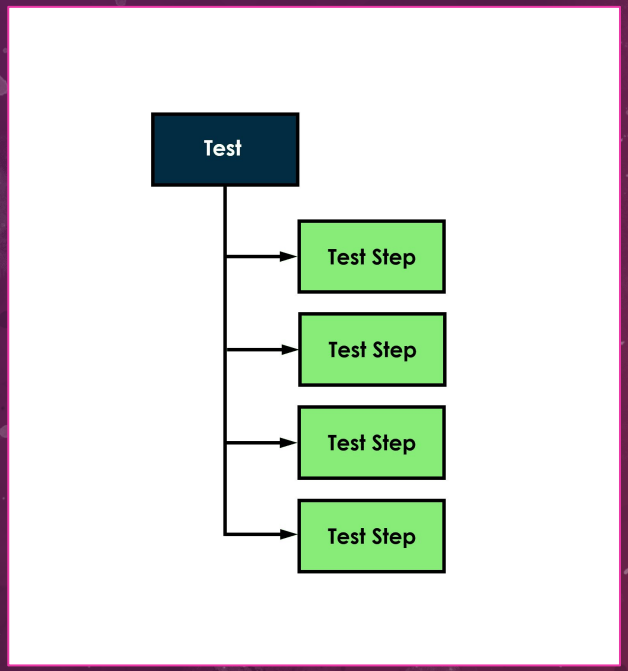
- Test suites
- Tests
- Test steps

Test suites

- A test suite groups tests together that are typically performed one after the other.
- Test suites represent a grouping of one or more tests or other test suites.
- The same test can be included within multiple test suites.
- However, a test suite can only be assigned one parent test suite.



Tests



"It represents an individual test case with a given acceptance criteria."

- An ATF test represents a set of actions and assertions to verify an expected end result.
- Each action or assertion within the test is represented by an individual test step.
- Although there is no limit to the number of test steps contained within a test, it is typically better to create smaller focused tests, rather than larger broad tests.

Test steps

- A test step represents a single action or assertion within a test, such as creating a user, opening a form, or validating a field value.
- Different types of test steps are created using different test step configurations.
- A test step configuration defines the behavior and characteristics of each type of action or assertion.