Continuous delivery

Andrey Zuev

Presentation Overview

- What is continuous delivery?
- Essential part of continuous delivery
- Summary

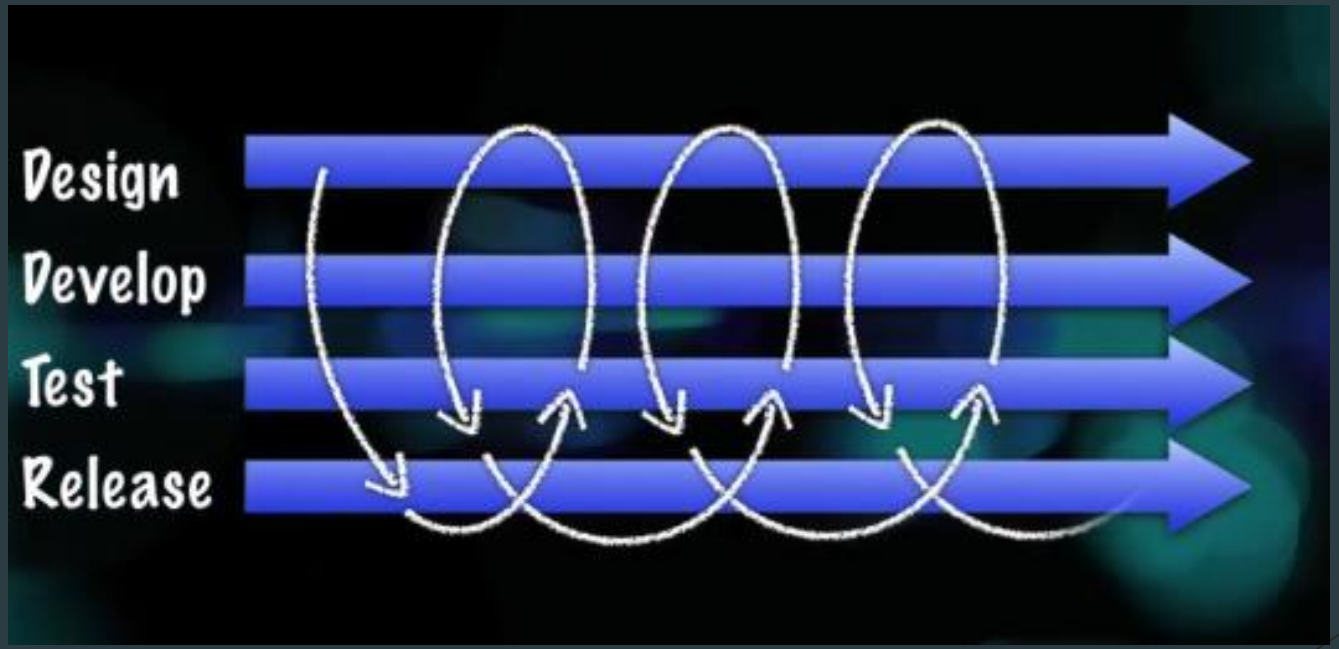
What is Continuous delivery ?

- It is a way of working so your software is always in a releasable state

What are the advantages of Continuous delivery

- 21% less time spent on unplanned work and rework
- 50% lower change-failure rates
- 50% less time spent fixing security Issues

Software activities



Efficient Feedback

- Software development is always about learning
- In order for Continuous delivery to work we need high quality quick feedback
- Continuous delivery is built on an idea of feedback

Strong engineering discipline

- Continuous delivery is a disciplinary approach
- Engineering is an application of scientific style reasoning
- Continuous delivery takes ideas of evidence seriously to achieve efficiency

Amount of work

Reduce the amount of work we do by two main approaches:

- Automation

- Process efficiency

This is grounded in the ideas of Lean Thinking

Continuous delivery

Continuous delivery is an application of:

- Scientific rationalism solving problems in software
- Lean thinking working efficiently and doing the most impact for the least amount of work

Cycle time and Lead time

- Cycle time is the interval of time between two consecutive deliverable to production.

- The amount of time a work item takes since it is first committed until it reaches production (time from “Commit” to “Release”)

Automation

- Automated Testing
- Automated Deployment
- Automated Configuration Management
- Automated Compliance

Deployment Pipeline

- Scope of Deployment Pipeline : A releasable unit of Software
- It is an idea to make a “Machine” that will take commit on one end and generate the releasable artifact in the end.
- Objective of Deployment Pipeline: To prove that his change is NOT fit for Production(we can do it by series of validation states)

Commit stage

- Ideally results < 5 minutes
- Objective: fail fast
- Objective: 80% Confidence

Acceptance test stage

- Test from the perspective of an external user
- Test life-like scenarios
- In a production-like test environment

Continuous delivery

- Test-driven development
- Continuous Deployment
- Continuous Integration

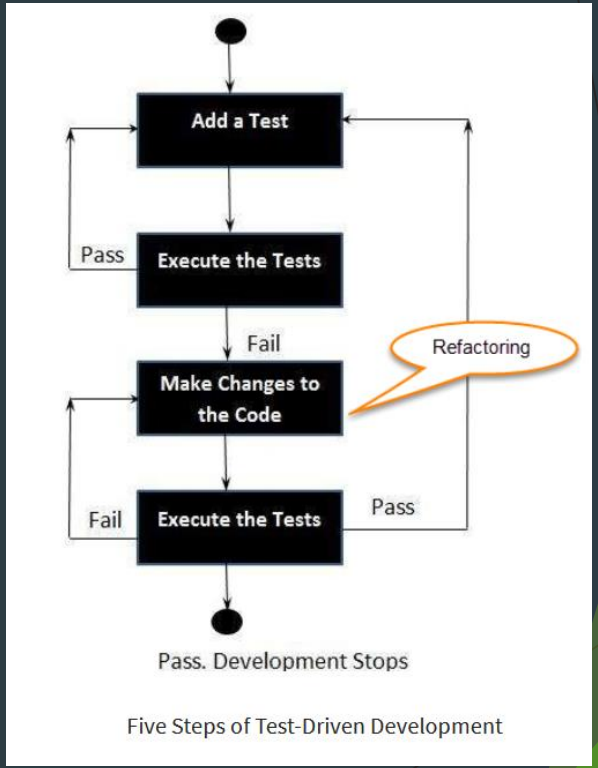
Test-Driven development

- **Test Driven Development (TDD)** is software development approach in which test cases are developed to specify and validate what the code will do. In simple terms, test cases for each functionality are created and tested first and if the test fails then the new code is written in order to pass the test and making code simple and bug-free

Test-driven development

Steps:

1. Add a test.
2. Run all tests and see if any new test fails.
3. Write some code.
4. Run tests and Refactor code.
5. Repeat.



Five Steps of Test-Driven Development

TDD advantage

- Early bug notification. Because we want to achieve about 80% of errors to be cut in 5 minutes of development, TDD perfectly fits the continuous delivery model.

TDD tools

- There are many tools to perform a test-driven development, but I think the most popular is Junit. I think everyone here who took Dr. Abdul's class is familiar with Junit so I will not go in detail on that.

Continuous Deployment

- Continuous Deployment (CD) is a software release process that uses automated testing to validate if changes to a codebase are correct and stable for immediate autonomous deployment to a production environment.

Continuous Delivery vs Continuous Deployment

- Think about receiving a package from your favorite online retail store. When waiting for a package to arrive you coordinate with a **delivery** service. This is the delivery phase. Once the package has successfully arrived, you open the package and review its contents to make sure it matches expectations. If it does not, it may be rejected and returned. If the package is correct, you are ready to **deploy**

Continuous Deployment tools

- Jenkins
- Gitlab
- Team city
- Bamboo

Continuous Integration

- The commit stage embodies continuous integration and takes it further
- You can not have a continuous delivery without continuous integration

Continuous Integration

- Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It’s a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository where builds and tests then run. Automated tools are used to assert the new code’s correctness before integration.

CI enable scaling

- CI enables organizations to scale in engineering team size, codebase size, and infrastructure. By minimizing code integration bureaucracy and communication overhead, CI helps build DevOps and agile workflows. It allows each team member to own a new code change through to release.

CI feedback loop

- Faster feedback on business decisions is another powerful side effect of CI. Product teams can test ideas and iterate product designs faster with an optimized CI platform. Changes can be rapidly pushed and measured for success. Bugs or other issues can be quickly addressed and repaired.

CI tools

- Jenkins
- Gitlab
- Team city
- Bamboo

Continuous Integration rules

- Always run commit tests locally before committing
- Always wait for a result
- Fix or revert failure within 10 minutes

Continuous Integration rules

- If a teammate breaks the rules revert their changes
- Ones commit passes move on to a next task
- If any test fails it is the responsibility of the committer

Summary

- Continuous delivery is a scientific rationalism solving problems in software, by applying test-driven development, continuous deployment and continuous integration

- Hopefully I encourage some of you to think about those methods and make software engineering process more efficient with a better quality


ANY
QUESTIONS?