Organization and control of continuous code delivery

Made by student
Ivanov O.A

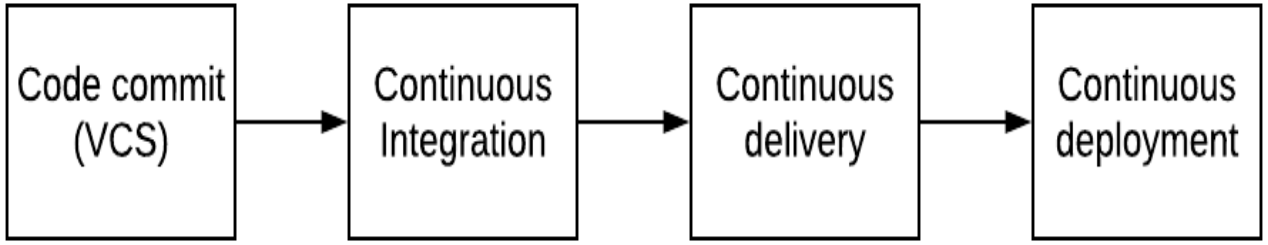
Introduction

- This course work were created to show how important CI/CD system and present modern strategies together with fundamental knowledge

Task formulation

- Provide definition of Continuous Integration.
- Describe approaches for Continuous Integration implementation.
- Describe common mistakes during implementation of CI.
- Uncover difference between Continuous delivery and deployment.
- Provide overview of release strategies. Uncover IaC principles.
- Design Zero-downtime approach using one of cloud providers.

The Problem of Delivering Software



Every time that you start project you should know the release date (the point in time then end user will be able to use your product). A lot of estimation, planning and other activities could be done before, but this date will be delayed due to bugs or undetected defects in software.

Common antipatterns

- Deploying Software Manually
- Deploying to a Production-like Environment Only after Development Is Complete
- Manual Configuration Management of Production Environments

CI

- Continuous integration was first written about in Kent Beck's book *Extreme Programming Explained* (first published in 1999).
- Continuous integration (CI) is the process of integrating new code written by developers with a mainline or "master" branch frequently throughout the day

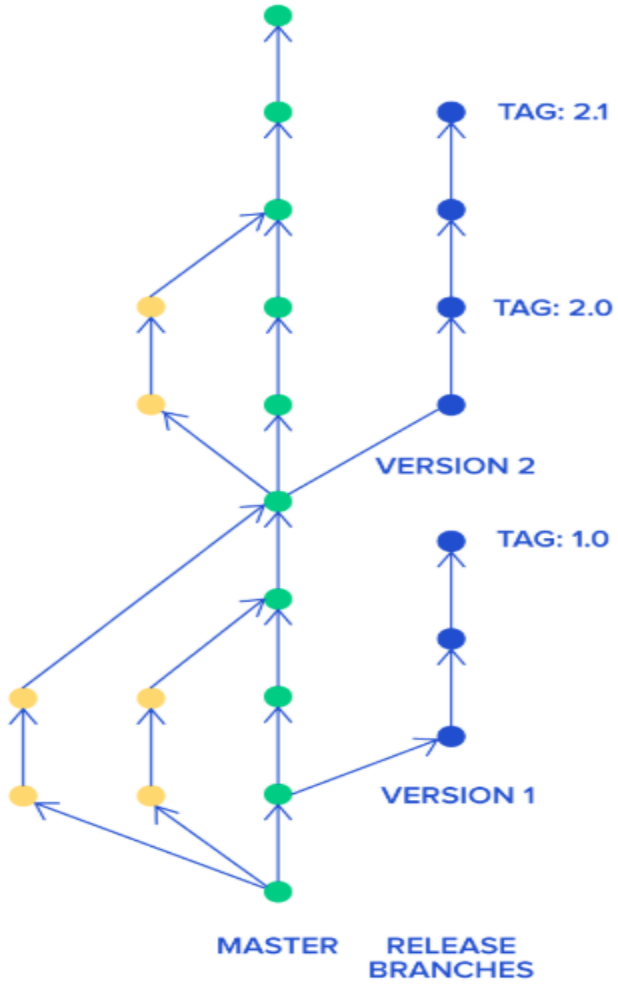
CI benefits

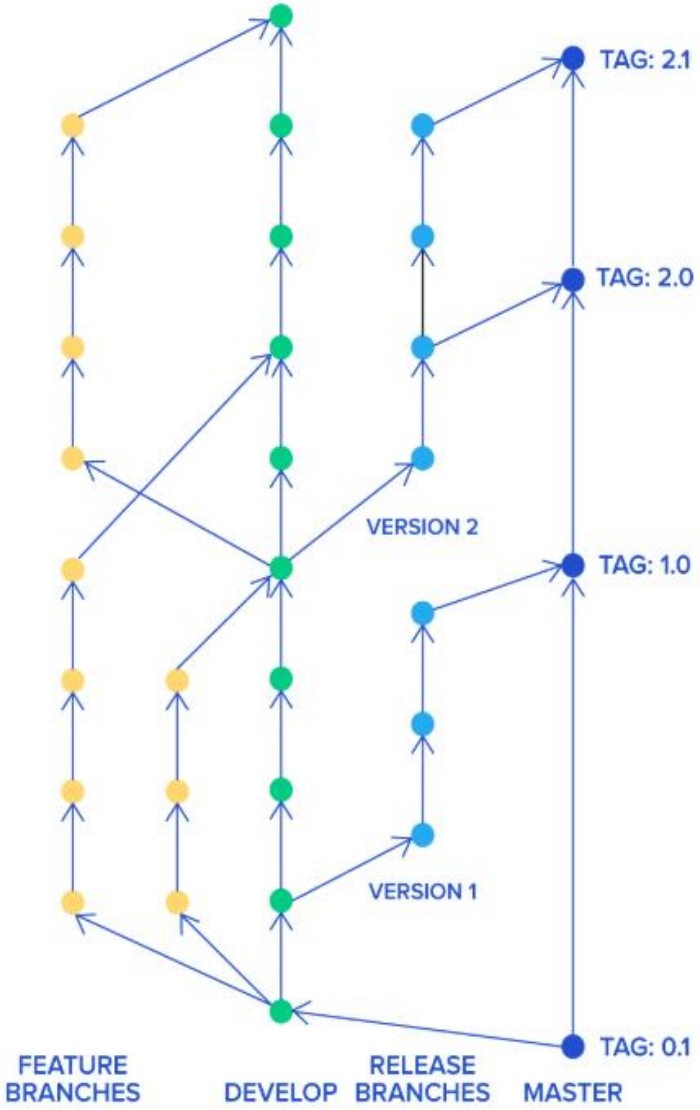
- Early Bug Detection
- Reduces Bug Count
- Automating the Process
- The Process Becomes Transparent
- Cost-Effective Process

Continuous Integration best practices by Martin Fowler

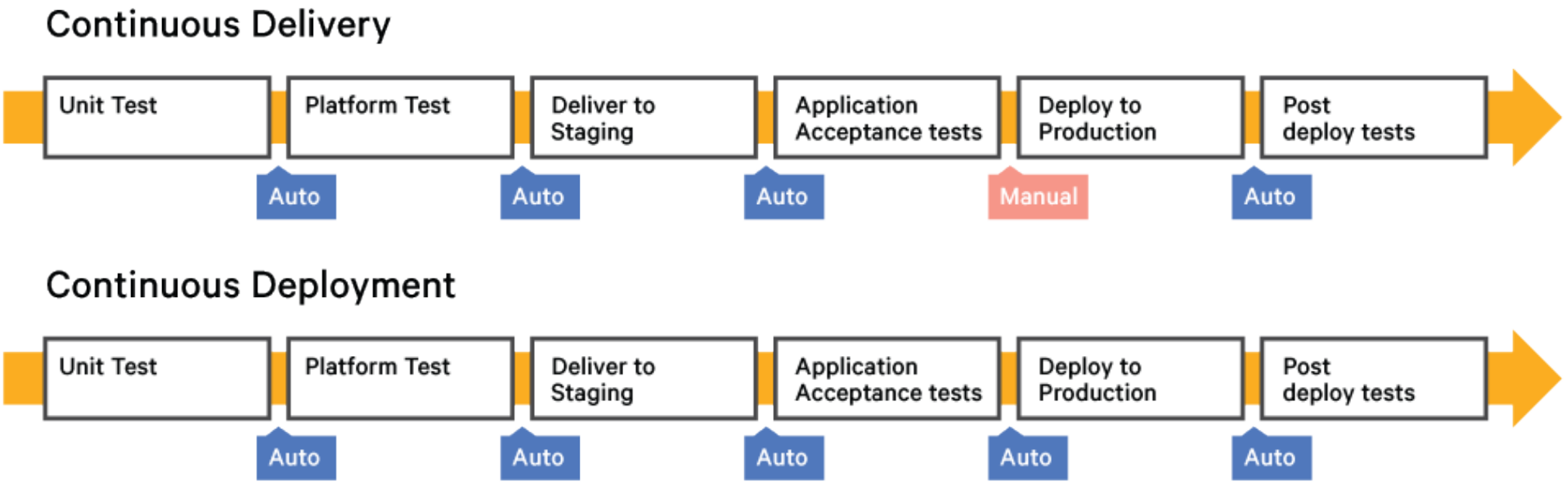
- Maintain a Single Source Repository
- Automate the Build
- Make the Build Self-Testing
- Everyone Commits to the Baseline Every Day
- Every Commit (to Baseline) Should be Built
- Keep the Build Fast
- Test in a Clone of the Production Environment
- Make It Easy to Get the Latest Deliverables
- Everyone Can See the Results of the Latest Build
- Automate Deployment

Branching strategies

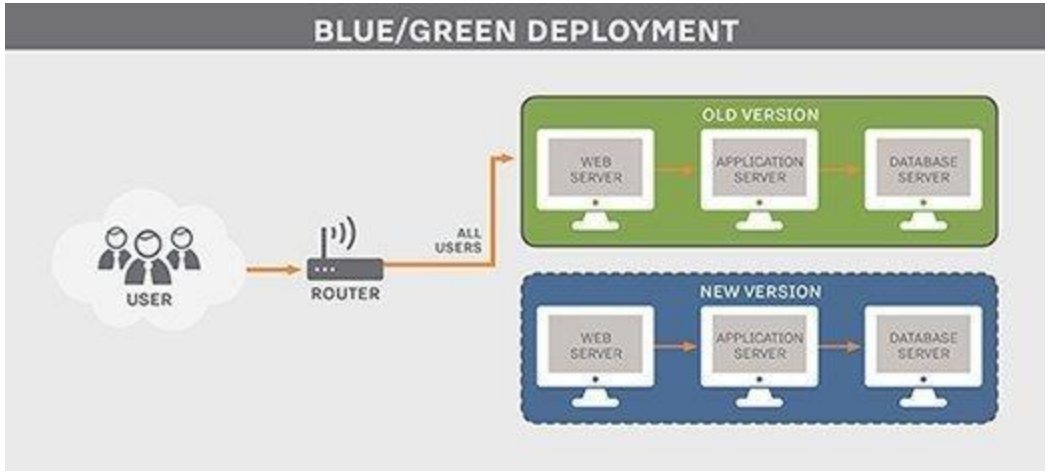


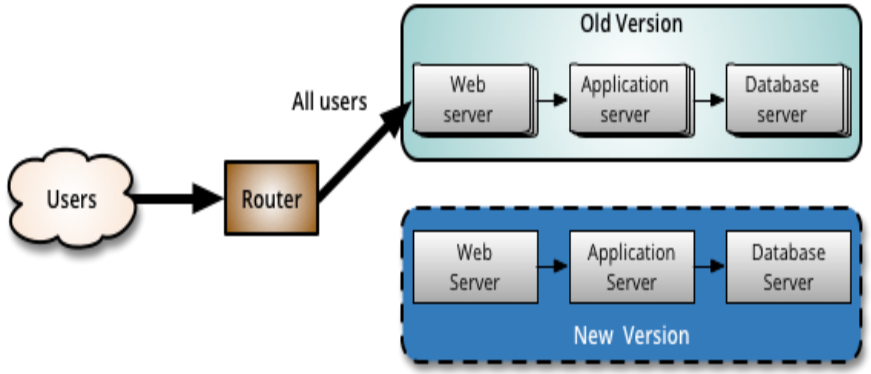


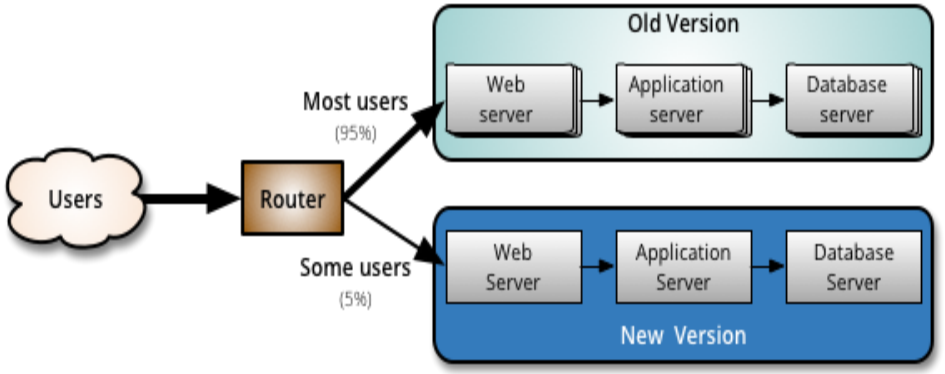
Continuous Deployment & Delivery

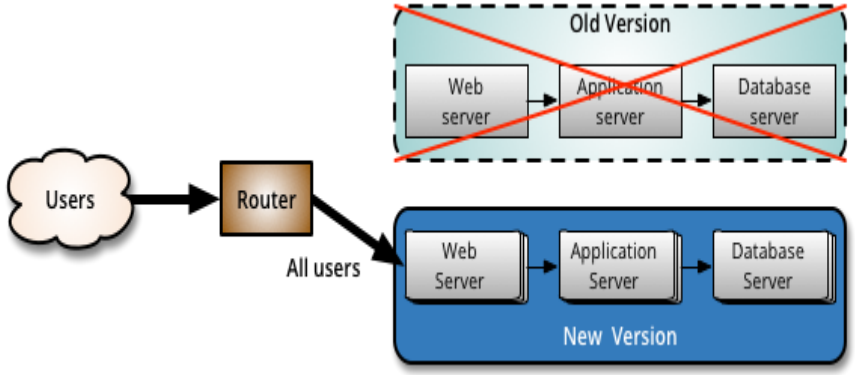


Release strategies





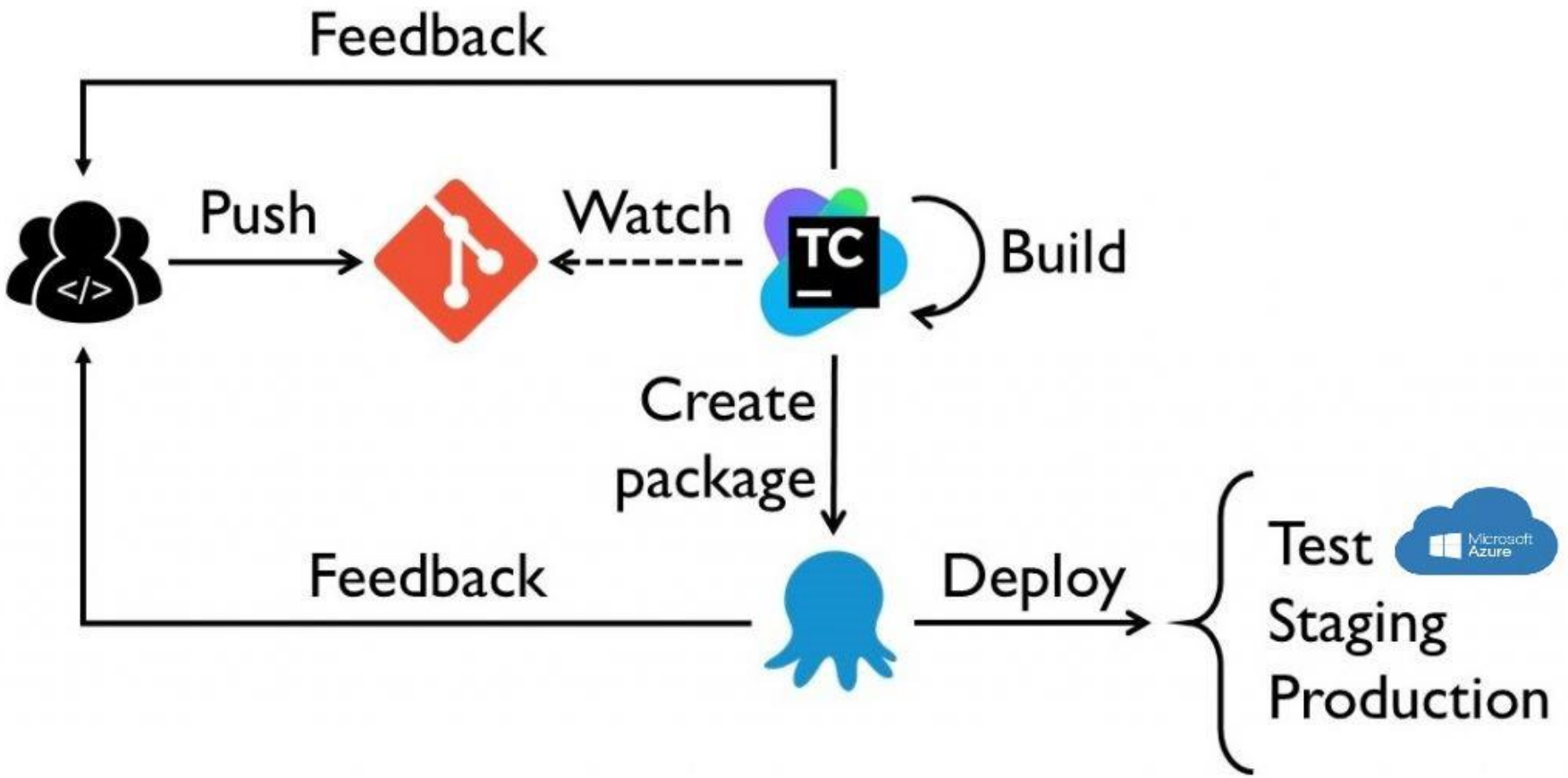




Infrastructure as a Code key points

- Systems Can Be Easily Reproduced
- Systems Are Disposable
- Systems Are Consistent
- Processes Are Repeatable
- Design Is Always Changing.

The general approach, that was used in real life project



Thank you