04834580 Software Engineering (Honor Track) 2024-25

DevOps

Sergey Mechtaev
mechtaev@pku.edu.cn
School of Computer Science, Peking University

Definition

DevOps is a collaborative and multidisciplinary effort within an organization to automate continuous delivery of new software versions, while guaranteeing their correctness and reliability. [1]

In 1994, Grady Booch stated [2] that in the context of iterative development,

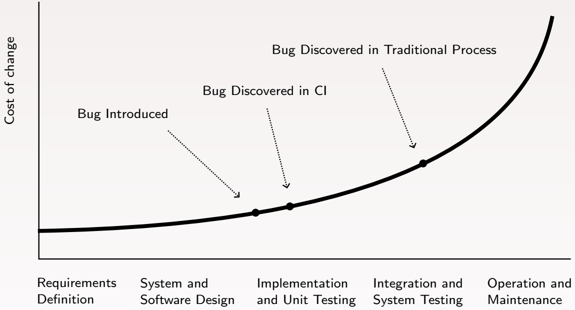
> *internal releases represent a sort of* **continuous integration** *of the system.*

Regarding the frequency of releases, he wrote that

> *for a modest-sized project, an organization may produce an internal release* **every two to three months**. *For more complex projects that require much greater development effort, this might mean a release* **every six months** *or so, according to the needs of the project.*

Continuous Integration

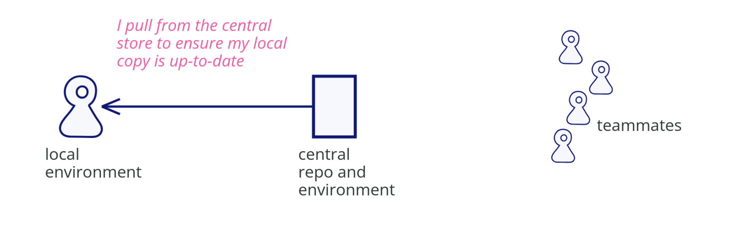
In 1998, Kent Beck (extreme programming) proposed frequent integration [3]: "code is integrated and tested **after a few hours** — a day of development at most."

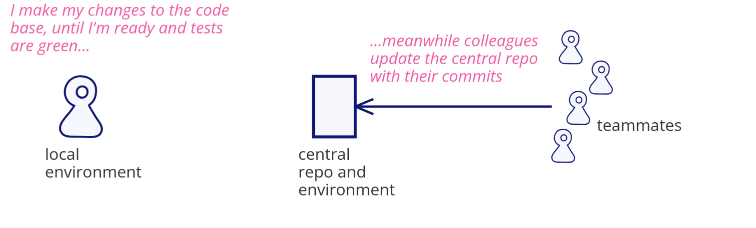


Kent Beck also mentioned another benefit:

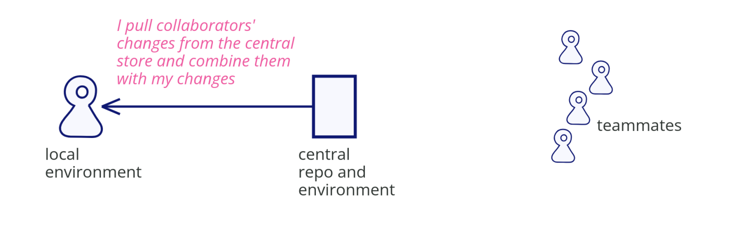
> *Integrating one set of changes at a time works well because it is obvious who should fix a test that fails — we should, since we must have broken it, since the last pair left the tests at 100%.*

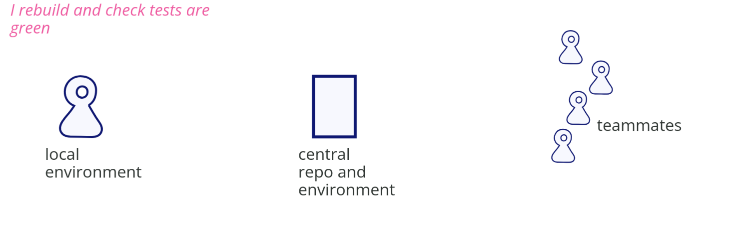
Continuous Integration Workflow [4]

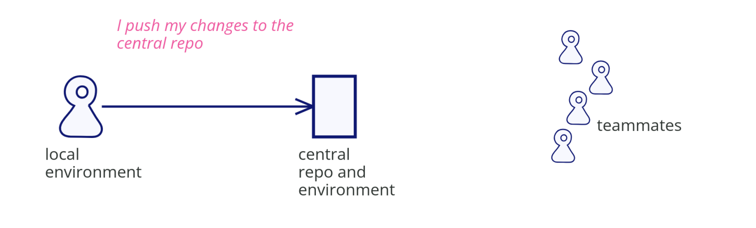


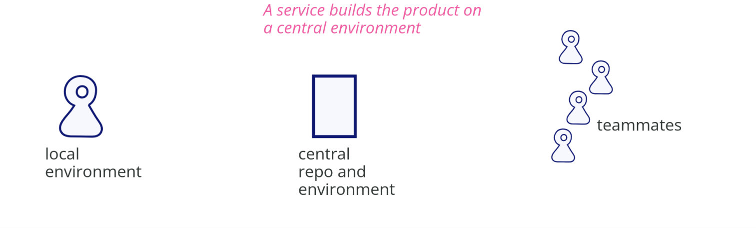


Continuous Integration Workflow

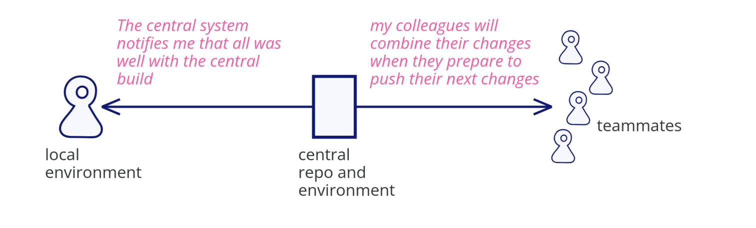





I push my changes to the central repo
local environment
central repo and environment
teammates


A service builds the product on a central environment
local environment
central repo and environment
teammates

Continuous Integration Workflow



Even if it "works on my machine", it may still fail on the CI service.

Continuous Delivery

Definition

A **deployment pipeline** is an automated implementation of your application's build, deploy, test, and release process. It is in essence the principle of continuous integration taken to its logical conclusion. [5]

Continuous Delivery: Why Not To Deploy Manually? [5]

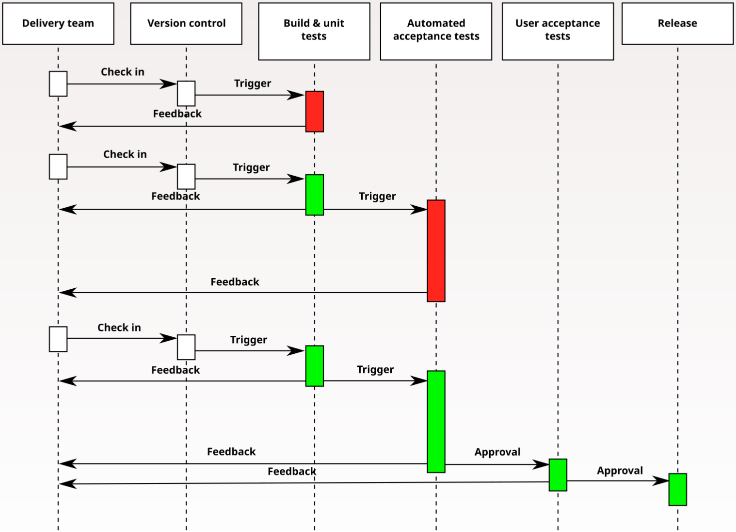
北京大学

- Without automation, errors will occur every time they are performed, time wasted on debugging deployment errors.
- A manual deployment process has to be documented. A set of automated deployment scripts serves as documentation.
- Automated deployments encourage collaboration, because everything is explicit in a script.
- Manual deployments depend on the deployment expert. If he or she is on vacation or quits work, you are in trouble.
- Manual deployments is boring and repetitive and yet needs significant degree of expertise.
- An automated process is fully auditable.

Continuous Delivery: Why Not To Deploy In The End? [5] 北京大学

- The longer the release cycle, the longer the development team has to make incorrect assumptions before the deployment occurs, and the longer it will take to fix them.
- In large organizations where the delivery process is divided between different groups such as development, DBA, operations, testing, etc., the cost of coordination between these silos can be enormous.
- The bigger the difference between development and production environments, the less realistic are the assumptions that have to be made during development.

Delivery Pipeline

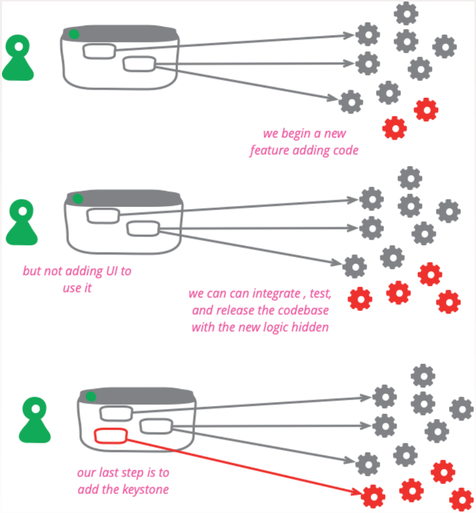


Continuous Delivery: Hiding Work-in-Progress [4]

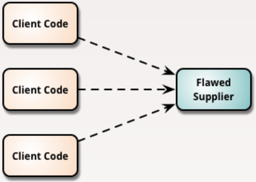
How to integrate/deliver before a user-visible feature is fully formed and ready for release?

- Keystone interface
- Dark launching
- Feature flags
- Branch by abstraction

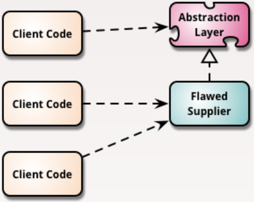
Keystone Interface [6]



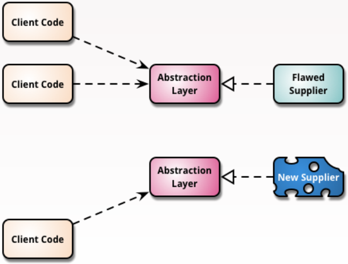
1.


Client Code
Client Code
Client Code
Flawed Supplier

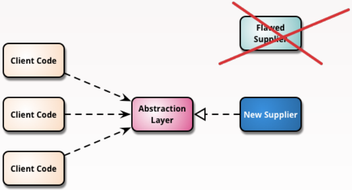
2.


Client Code
Abstraction Layer
Client Code
Flawed Supplier
Client Code

3.

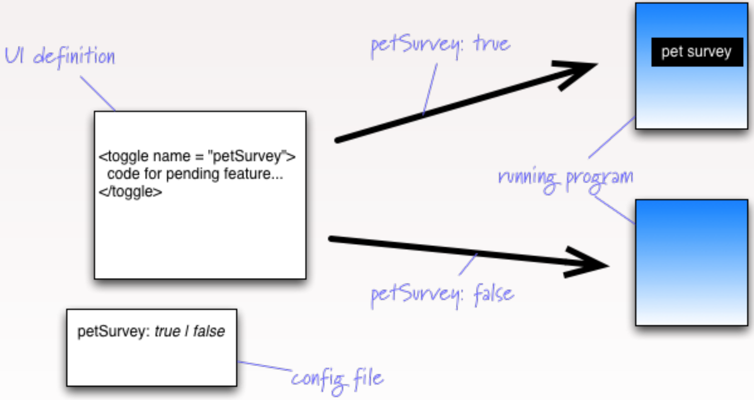

Client Code
Client Code
Abstraction Layer
Flawed Supplier
Abstraction Layer
New Supplier
Client Code

4.


Flawed Supplier
Client Code
Client Code
Abstraction Layer
New Supplier
Client Code

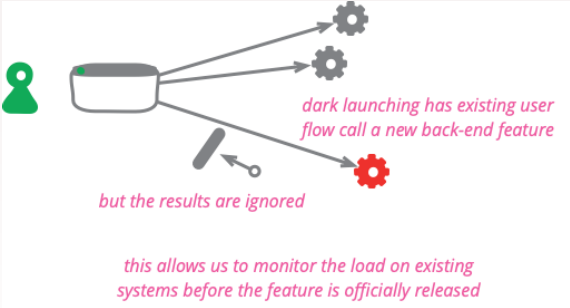
Feature Flag [8]

The running application then uses future flags from configuration in order to decide whether or not to show the new feature.



Dark Launching [9]

Dark launching a feature means taking a new or changed back-end behavior and calling it from existing users without the users being able to tell it's being called.



Gitflow [10] is a branching model proposed by Vincent Driessen in 2010.

Main branches:

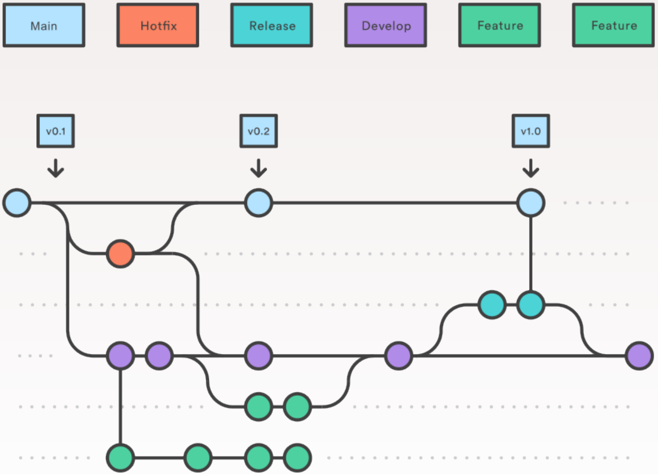
main reflects a production-ready state.

develop reflects a state with the latest development changes for the next release.

Supporting branches:

- feature branches to develop new features for the upcoming or a distant future release;
- release branches to support preparation of a new production release;
- hotfix branches for unplanned releases.

Gitflow

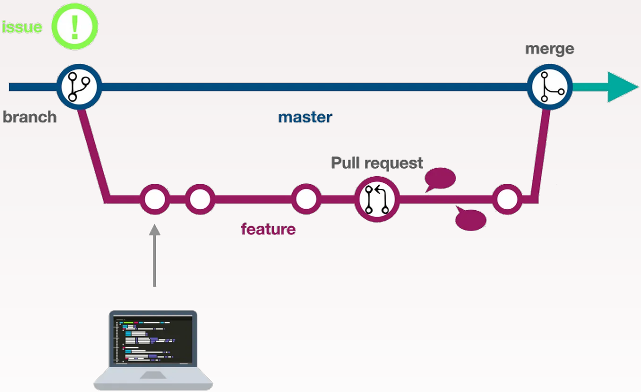


GitHub Flow

GitHub Flow [11] is a simplified Gitflow:

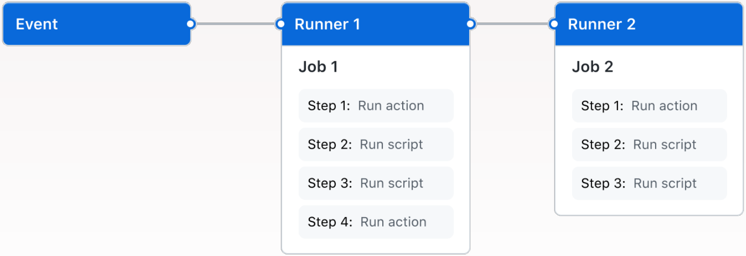
- Anything in the main branch is deployable.
- To work on something new, create a descriptively named branch off of master (ie: `new-oauth2-scopes`).
- Commit to that branch locally and regularly push your work to the same named branch on the server.
- When you need feedback or help, or you think the branch is ready for merging, open a **pull request**.
- After someone else has reviewed and signed off on the feature, you can merge it into **main**.
- Once it is merged and pushed to main, you can and should deploy immediately.

GitHub Flow



GitHub Actions is GitHub CI/CD service.

- **workflow** to be triggered when an **event** occurs in your repository
- **workflow** contains one or more **jobs**
- each **job** will run inside its own virtual machine **runner**
- each **step** either runs a script or an action (a reusable extension)



Defined in `yaml` files:

```yaml
name: learn-github-actions
run-name: ${{ github.actor }} is learning GitHub Actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install -g bats
      - run: bats -v
```

北京大学

[1] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles.
A survey of devops concepts and challenges.
ACM computing surveys (CSUR), 52(6):1–35, 2019.

[2] Grady Booch, Robert A Maksimchuk, Michael W Engle, Bobbi J Young, Jim Connallen, and Kelli A Houston.
Object-oriented analysis and design with applications.
ACM SIGSOFT software engineering notes, 33(5):29–29, 2008.

[3] Kent Beck.
Extreme programming explained: embrace change.
addison-wesley professional, 2000.

[4] Martin Fowler.
Continuous integration.
`https://martinfowler.com/articles/continuousIntegration.html`, January 2024.

[5] Jez Humble and David Farley.
Continuous delivery: reliable software releases through build, test, and deployment automation.
Pearson Education, 2010.

[6] Martin Fowler.
Keystone interface.
https://martinfowler.com/bliki/KeystoneInterface.html, 2020.

[7] Martin Fowler.
Branch by abstraction.
https://martinfowler.com/bliki/BranchByAbstraction.html, 2014.

[8] Martin Fowler.
Feature flag.
https://martinfowler.com/bliki/FeatureFlag.html, 2010.

北京大学

[9] Martin Fowler.
Dark launching.
https://martinfowler.com/bliki/DarkLaunching.html, 2020.

[10] Vincent Driessen.
A successful git branching model.
https://martinfowler.com/bliki/DarkLaunching.html, 2020.

[11] GitHub Developers.
Github flow.
https://martinfowler.com/bliki/DarkLaunching.html, 2020.