Optimize App release circle by applying CI/CD

It’s worth a try

Intro

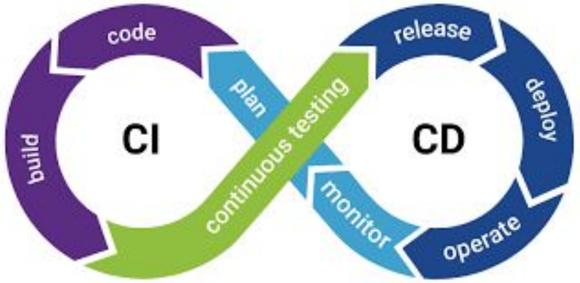
Continuous integration & Continuous deployment (CI/CD) have several benefits, but I only mention some of them which can deal with the most painful problems of my imaginary company: An E-Commerce company which offer its service via website and mobile app.

Definitions

Continuous integration (CI): A practice of merging all developers works in a shared mainline several times a day.

Continuous deployment (CD): An approach practice in which value is frequently delivered through automatic deployment

CI + CD = Continuous delivery: A practice in which teams produce and release value in short cycles



Definitions

To implement CI/CD, requires

- Budgets for CI/CD tools
- Change in working mindset and culture

=> Set **Continuous delivery** as the True North

Let’s see what CI/CD can helps current business!

Business context (problem 1)

Avg. duration for a project from the idea to production takes at least 2 months

New ideas come frequently almost every 2 weeks

-> Force developers to work in separate projects

=> Solving code and environment conflicts take many time & resources, leads to delay of release plan

=> New ideas are not “new” anymore, competitors have introduced them first.

Business context (problem 2)

New update frequency: every 2 weeks

Avg. deployment time (include smoke test): 2 hours

Avg. roll back and recover time: 4 hours

These things are being done manually

=> Error prone, may leads up to 6 hours of down time. Some users are complaining and giving low rating on app stores

Business context (problem 3)

Test cases in dev and staging env passed, but incur issue in productions, because each environments have some unawared difference in setting

=> Negatively impact customer experience, take time to troubleshoot problem and feedback

Solution

Problem 1:

Using CI/CD, update to production can be delivered frequently, automatically

-> Project can be teared down into minor updates, and still can generate profit...faster => more profit while reduce deployment pressure

-> Developers and business team can quickly evaluate the project in production,can quickly withdraw if not effective

=> Saving costs by quickly cutting down ineffective projects

Solution

Problem 1:

All team members are on the same page thanks to frequent deployment

-> minimize gaps and conflicts between developing projects

=> reduce cost

Solution

Problem 2:

To implement CI/CD, all phases must be automatic, including testing and deploying

-> Reduce human error, more consistent and reliable

=> Minimize downtime and rollback, make app more reliable, increase users rating => reduce manforce cost & earn more profit

Solution

Problem 3:

Systems are built and configured by code, which can be versioned, copied and applied among many environments

-> Ensure consistency between environments, all issue can be detected before hand in DEV before promoting to Production

=> Avoid risk of deploying faulty app version, reduce cost

Solution

Problem 3:

Auto test cases can be reused in many environments

-> Developers have more time to focus building new features, rather than testing and fixing bugs

=> Reduce cost, strengthen competitive advantages