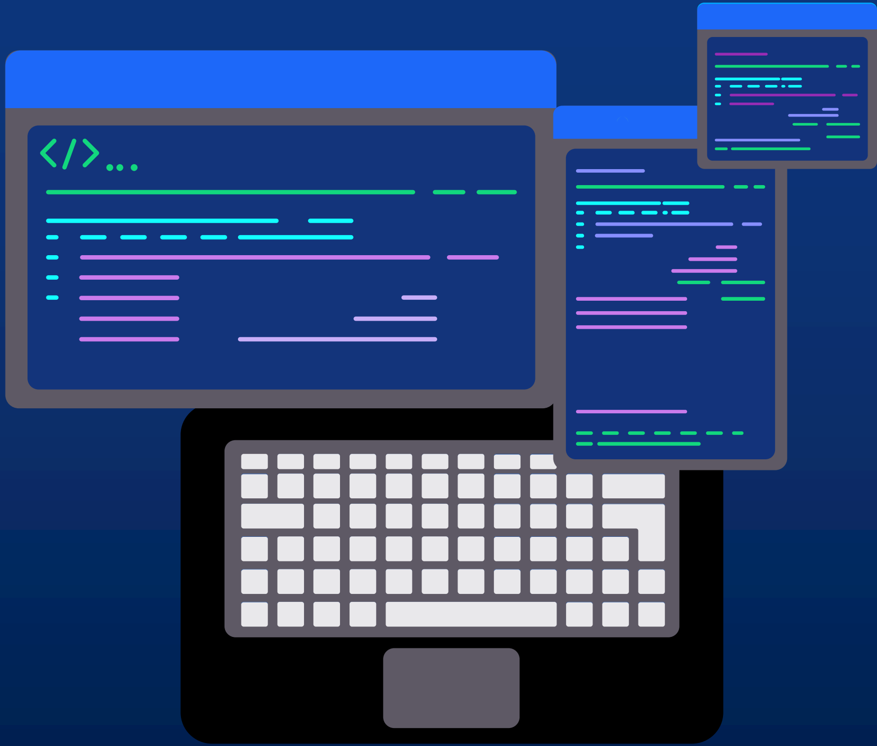
CONTINUOUS DELIVERY

AN EFFICIENT WAY TO DELIVER SOFTWARE



Integration and Deployment Issues We Face After A Sprint

Currently, our strategies and processes for development to delivery are not optimum.

- We wait too long to push a **new feature** to production since we usually wait for all new features to be pushed together.
- We get too many **merge conflicts** from our developers when they release a large chunk of code they have been working on for a very long. Thus, we are unable to meet our deployment timelines.
- Our testing process takes too long since our developers spend a lot of time looking for **anomalies** in the main branch.

How Can Continuous Delivery Improve our Product?

We can do this through CI/CD

CI

Continuous Integration

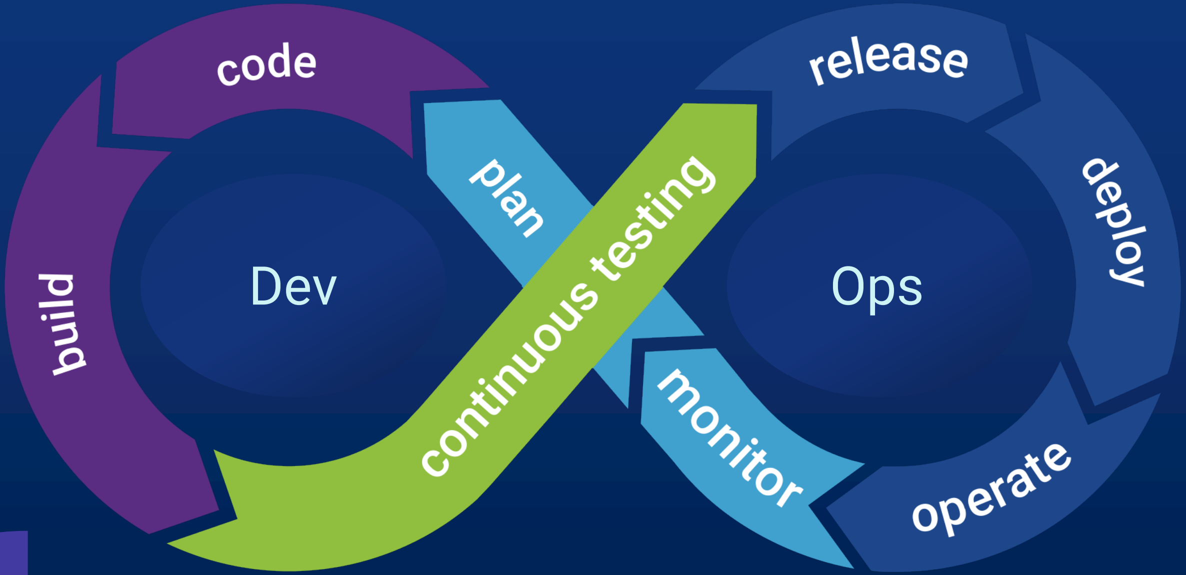
This is a practice of merging all developers’ working copies to a shared mainline several times a day

CD

Continuous Deployment

This is a software engineering approach in which the value is delivered frequently through automated deployments

CI/CD in a Nutshell



What We Can Achieve With CI / CD In Terms Of Cost, Revenue, Profit

- Automated Smoke Testing : This would safeguard our revenue through a reduction in downtime brought by crashes and errors from deployment.
- Catching Unit Test Failures Faster: We could save money if our production software included fewer bugs and we spent less time performing manual testing.
- Faster and Frequent Deployments to Production: We would get more revenue by pushing value generated features more frequently to our consumers.
- Less human error and merge conflicts: Through automating our infrastructure we could reduce our server expenditure since more servers will spun up when there is a lot of traffic and spun down when traffic is low. Also, since deployments would be made frequently, merge conflicts would reduce thereby saving time and money.