
CI/CD

What is happening now without CI/CD?

1. Budget is spent on non-deliverable work

- The more time our engineers spend performing tasks to get their code into production is less time spent creating deliverable work for clients
- This is even more prevalent in smaller teams where our lead engineers usually become the go to people to resolve integration and deployment issues, which is valuable time away from development

2. Delayed Delivery

- When thee are too many dependencies across teams and manual tasks delays and mistakes inevitably come into play.
- This results is net new features for clients being delayed as well as any bug fixes being implemented asap.

3. Lower developer productivity & retention

- Developer hiring and retention will inevitably suffer as a result of developers focus being away from code related to business logic.

What are the benefits of good CI/CD?

1. Increased speed of innovation and ability to compete in the marketplace

- The more automation involved in the intgegration and deployment of code means faster turnaround times to get new business ideas into the hands of client

2. Separation fo deployments from releases

- With good automation we can facilitate business stakeholderto rollout features when they are ready and teston a group of users prior to rollout

3. Ability to attract and retain talent

- Engineers that can focus on what they're best at will be happier and more productive, and that has a far-reaching impact. Turnover can be expensive and disruptive. A good CI/CD strategy means engineers can work on important projects and not worry about time-consuming manual tasks. They can also work confidently knowing that errors are caught automatically, not right before deployment. This kind of cooperative engineering culture inevitably attracts talent.

4. Higher quality code and operations due to specialization

- Dev can focus on dev. Ops can focus on ops. Bad code rarely makes it to production because testing is automated. Developers can focus on the code rather than the production environment, and operations doesn't have to feel like a gatekeeper or a barrie

How would you measure success?

1. Cycle time

- Cycle time is the speed at which a DevOps team can deliver a functional application, from the moment work begins to when it is providing value to an end user.

2. Time to value

- Once code is written, how long before it's released? This delay from when code is written to running in production is the time to value, and is a bottleneck for many organizations. Continuous delivery as well as examining trends in the QA process can help to overcome this barrier to quick deployments.

3. Uptime, error rate, infrastructure costs

- Uptime is one of the biggest priorities for the ops team, and with a good CI/CD strategy that automates different processes, they should be able to focus more on that goal. Likewise, error rates and infrastructure costs can be easily measured once CI/CD is put in place. Operations goals are a key indicator of process success.

4. Team retention rate

- Developers who are able to focus on their primary role, writing code, are much more likely to stick around.
- Building a time with a high retention ate allows us to delivery bigger andbetter features as we have a team with years of experience in our product and product domain.