Using CI/CD to improve our development workflow

Situation

Currently the development team works on a large Epic and then follow a set of deployment steps in order to deploy the ready features to production for our users to see. This process is riddled with a couple of problems:

- Product development takes long
- Our deployment process involves a lot of human steps which are prone to errors. A single error in the deployment process would result in a botched deployment.
- New features get released a lot slower to our users.

Proposed Solution

CI/CD

Moving to adopting Continuous Integration and Continuous deployment would allow the company save costs in developer hours/development time and also increase revenue by reducing the time it would take for features to get to our end users.

Key Benefits

1. Faster release rates.

CI/CD would enhance the rate at which we deliver features from our product backlog, because we would have a pipeline to support automated testing of the code on each integration with new code. This ensures that no new features break existing ones and we can focus on delivering newer features without worrying about whether or not we have introduced regressions. This means our development team spends more time building newer features than debugging a botched deployment.

2. Customer satisfaction

With CI/CD we would be able to focus on delivering value to the customer at a much quicker rate. Our CD pipelines would enable us roll out new features almost seamlessly and consistently every time (with minimal human intervention). This leads to reduced cost in terms of deployment/debugging time and increased revenue in terms of users getting value much quicker.

3. Smaller code changes

A well defined CI process would mean that the team wouldn't need to chunk up large features waiting for one deployment day. Instead we would have smaller well defined features for developers to work on, which can then be quickly finished and integrated into the main codebase and deployed. This results in less bugs in production and we can avoid the costs incurred as a result.

4. Reduced defects in deployed code

A key part of our CI/CD pipeline involves the use of tests to guard against defects or regressions in our code. Any defects introduced to the codebase would be caught early and can be resolved quickly. With this we would have less bugs in production and automated tests would have us spending less time testing for regression.