Continuous Integration (CI)/Continuous Deployment (CD) in UdaPeople

Mohammed Isioye

What is CI/CD?

Continuous Integration is a practice that is common to developers who are working daily to achieve the purpose of building a product to solve a problem. This practice has a lot to do with developers working individually (technically) towards the goal of the team. This practice enables them to contribute to a common pipeline to achieve this goal

Continuous deployment is a software development practice in which product milestones are delivered frequently through the process of automation.

Continuous delivery is process of employing both continuous integration and continuous deployment into software development life cycle

Implementing CI/CD at UdaPeople

In order to achieve Continuous Integration, Deployment and Delivery most of what our developers normally do has to be automated in such a way that it's fool proof, that is, except for a change in process, the automation will be solid enough to handle tasks the usually required physical presence like deploying new features to production for our customers to try out and give feedback, test new features before it's delivered to our customers etc. The following are some of the steps required to be automated

- Static code analysis
- Unit testing
- Smoke testing
- Deploying new feature to production
- Creating infrastructure that our product requires to scale successfully
- Handling reversion in case things go wrong

Why CI/CD at Udapeople

- **Product delivery**: CI/CD will help us to deliver our product to customers in time and in optimal state. Also the speed and efficiency of adding new features is better managed when CI/CD is adopted. This is so because the feedback loop when using CI/CD is faster when building products.
- **Code Quality**: Smoke testing is a crucial part of CI/CD and this enforces code quality. The amount of technical and financial resources needed to manage downtime/failures will reduce drastically.
- **Cost Reduction**: With CI/CD, the quantity of bugs/failures reduce which increases the time engineers will spend on adding new features and functionalities to our product
- **Bug and Incident management**: When smoke tests fails and they will as a result of new deployments which may break production; it is easier to manage bug incidents because the development team will get alerts when things go wrong which will make bug management easier and faster which in turn reduces the risk of financial loss due to technical downtimes.

Thank you!