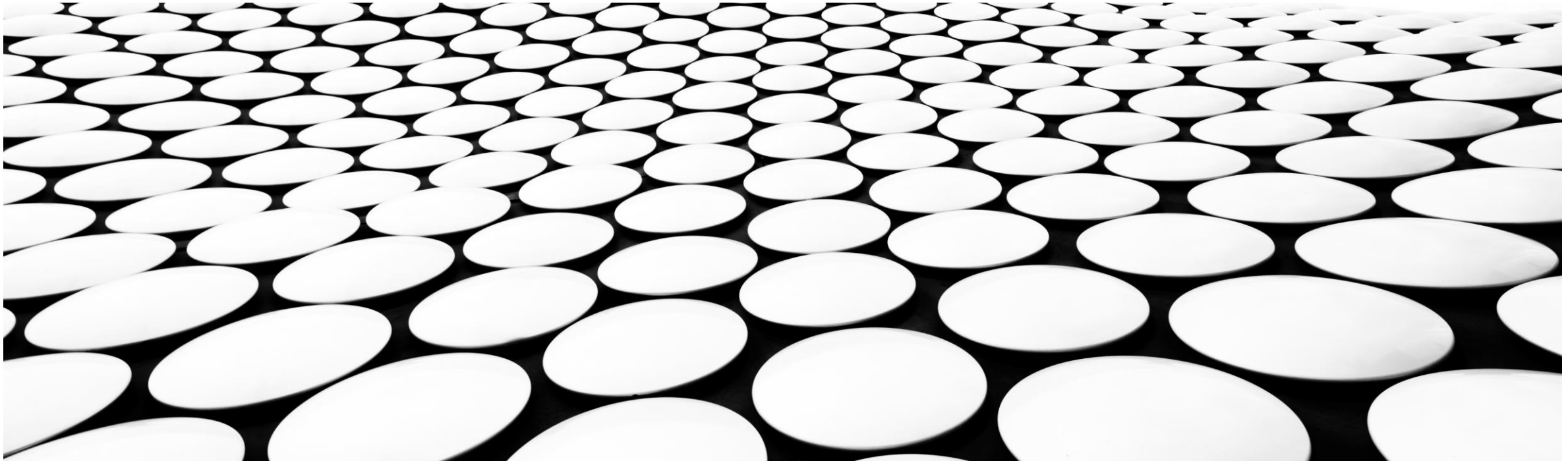
CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY OR DEPLOYMENT

THE FUTURE OF AUTOMATION – COSTS, BENEFITS AND WHY WE MUST ADOPTS CI/CD



WHAT IS CI/CD?

➢ CONTINUOUS INTEGRATION

Think of Continuous Integration as a process of automating the building and testing code.
A successful CI means developers are able to frequently make code changes to an app, test it and merge it across shared repository.

➢ CONTINUOUS DELIVERY

In continuous delivery code changes are tested for bugs, packaged for rollout to production automatically.

➢ CONTINUOUS DEPLOYMENT

This builds on from the benefits of continuous delivery, by automatically releasing a developer's changes from the repository to production after successful evaluation, where it is usable by customers.

HOW DO WE FIT IN?

According to a 2020 Survey by GitLab posted on their blog gives a full picture how this fits into Udapeople vison which promises to help small businesses care better for their most valuable resource: their people. In the survey 83% of developers said they're releasing code faster than ever before. Nearly 60% of them deploy codes multiple times a day, once a day, or once every few days (that's 15 percentage points higher than in 2019). While between 2019 and 2020 about 21% of developers said their teams had added CI to their process, while just over 15% brought in continuous deployment. For the Operations Team nearly 40% said their development lifecyle is "mostly" automated, Over half of them are managing cloud services, while 42% said they're now primarily managing hardware and infrastructure.

A surprising benefit of CI/CD: Changing development roles | GitLab

COST IMPLICATIONS AND BENEFITS

What we did?	Implications	Moving Forward	Benefits
Manual Testing of codes	Developers spends more time trying to fix issues.	Automated Testing-	Avoid Cost, Developers are able to catch bugs faster so Less bugs get shipped to production
Manual merging of code and pushing to staging and then production	Difficulty in catching compile errors after merge	Automating CI	Reduce Cost, free up developer's time, enabling them to focus more on product development.
Release cycle begins with Change Management (CM) tickets	Excessive coordination, and Longer Product Release Time	CI/CD to continuous merge codes and continuously deploys them to production	Increase Revenue due to faster product Release Rate.

COST IMPLICATIONS AND BENEFITS

What we did?	Implications	Moving Forward	Benefits
Lack of centralized monitoring	We get feedbacks on problems in a production release from users.	Automated Rollback Triggered by Job Failure	Protect Revenue by rollback, as we are able to rollback to a working state of service and prevent production downtime.
single shared integration environment	If one person or team broke the backend service, testing would come to a halt for all the other teams.	Automating CI/CD using Deployment Pipeline throughout development, testing, production, and monitoring phases of the software development lifecycle.	Increase Revenue, as New value-generating features will be released more quickly.