CI/CD presentation

CI/CD preview

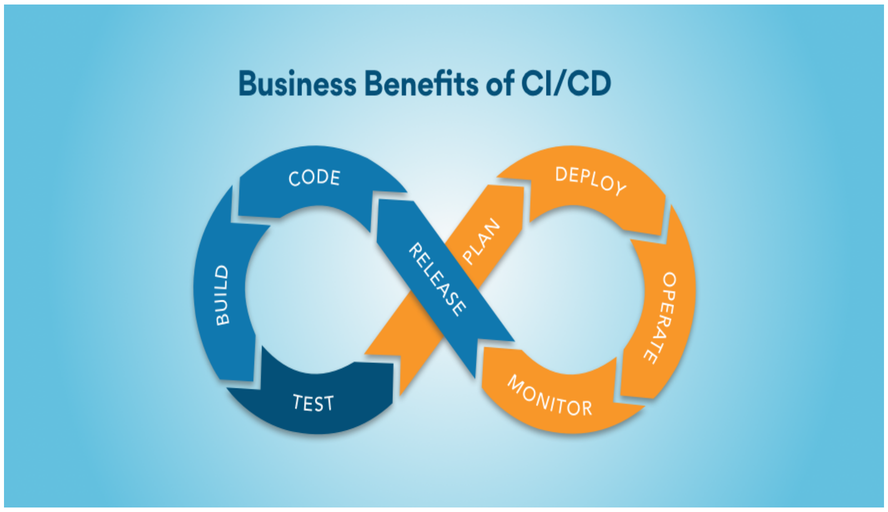
Good morning Sir, I just need to discuss with you this proposal which will give our company the ability to reduce costs, reduce time, create revenue and protect revenue .

- Why CI/CD

1. With CI/CD we will have the capability to **deliver software changes to users in a timely**, repeatable, and secure manner by **introducing automation into software development processes**.

2. **From merging codes to testing builds,** continuous integration (CI) validates all the stages of the development process while optimizing the code release cycles through automation capabilities. This, in turn, **minimizes the probability of extended feature development cycles** and **related issues like merge conflicts**.

3. **Continuous deployment** (CD) **focuses on setting up a bundled artifact into a production environment** in the fastest way possible. It automates the whole distribution process, including deployment.



- Current state

Whe are facing a lot of problems now :

1. **Shipping code was a painful process**. A release cycle took weeks to complete. Simple **bug fixes took a laborious two-week** route to production, and the releases themselves were fragile and inevitably introduced new bugs. **On top of that, the manual process of deploying often failed.**
2. The accumulated complexity and fragility meant that **excessive coordination was needed: 30 people in a room**, eight days before each deployment, **figuring out the right order of dependencies for 12 code bases**, depending on the nature of what was contained in a given release.
3. There was a **single shared integration environment for testing** any changes, which meant that **if a database or backend service broke in some way**, testing would come to a screeching halt for all other teams.
4. Builds were packaged up as **RPMs to make system-level changes in the virtual-machine based deployment environment**, even as the rest of the world moved on to containerization.
5. **There were no automated integration tests** covering our consumer experience, **so testing was a manual process that took place in a single QA environment over the course of a few days.**
6. **Once QA signed off on a release**, a member of the production engineering team would manually deploy the RPMs to production in the order determined during the coordination meeting.
    - After all of that we need to be faster and reduce the time with cost, because of that i will talk about CI/CD.

Future state

- What will we gain with this proposal

- **Higher efficiency**

Increased productivity is one of the leading advantages of a CI/CD pipeline. We should automate your process if we have a review process that includes deploying code to development, testing, and production environments and entering multiple commands across several domains. This creates the need for a CI/CD framework.

- **Reduced risk of defects**

Finding and resolving defects late in the development process is costly and time-consuming. This is particularly true when problems arise with features already released to production.

We can test and deploy code more frequently using a CI/CD pipeline, giving QA engineers the power to identify and fix errors as soon as they occur. **This way, We are effectively mitigating risks in real-time**.

- **Faster product delivery**

With a smooth CI/CD workflow, multiple daily releases can become a reality. **Teams can automatically build, test, and deliver features with minimal manual intervention**. **Docker**, **Kubernetes** are some of the tools and frameworks that can be used to accomplish this.
CD enables our team to provide customers with frequent and timely updates. When CI/CD is used, **the entire team's efficiency increases**, including the release of new features and fixes to problems. **Businesses can address market shifts**, security challenges, consumer needs, and financial pressures faster.

- **Log generation**

Observability is pivotal for DevOps. If something isn't right, you need to figure out why. We will need a way to track the system's performance over time to determine essential performance indicators. **Observability is a technical tool that aids in this endeavor**.
Logging information plays a vital role in observability. Logs provide a large volume of information to decipher what's happening beneath the UI and study program behavior. **A CI/CD pipeline generates a lot of logging data at every level of the.**

- **Quick rollback if required**

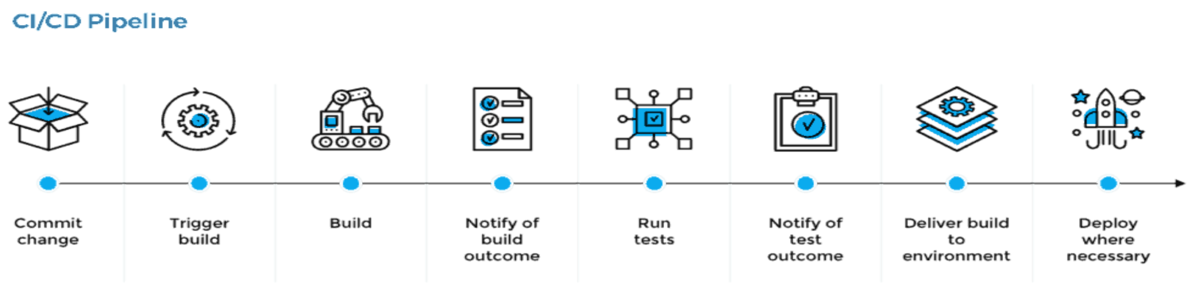
One of the most exclusive benefits of a CI/CD pipeline is that it leads to the **quick and easy rollback of code changes if there are any issues in the production environment after a release**. If any new code change breaks a feature or general application, **We can revert to its previous stable version right away**. We can deploy the most recent successful build instantly to avoid production interruptions.

- **Better planning**

Organizational designs must be adaptable to changing economic conditions. However, it's difficult for development and testing teams to adapt to rapid changes in dynamic business conditions. **A CI/CD pipeline enables organizations to accomplish this by ensuring that they have a well–organized surplus of items and a continuous line of communication with clients.**

- **Cost-effectiveness**

The CI/CD pipeline takes a different approach to software delivery. It can be compared to an assembling unit's delivery pipeline. In any business situation, time and assets are essential. **Firms are expected to respond to client demands quickly and effectively with such requirements.**



- **What Actions we need to move to CI/CD**

1. **The Review Phase**

WE can't jump headlong into a CI/CD project without knowing exactly where our starting point is. In the review phase, **we need to kick off by understanding the requirements of the different stakeholders involved**. And in doing so, it's better to get this down on paper and document the varied and inter-relating goals that different people may have of the CI/CD project.

2. **The Define Phase**

In the Define phase, this is, fairly obviously, where we define what our CI/CD implementation will look like. If we do not have CI/CD expertise within our organization.

Whether we have internal or external expertise, we will need to develop a new design flow for your releases and Continuous Integration. That will include the branching and merging strategy and the execution of test automation. When that is understood, and not before, it is time to make recommendations on the CI tools and technologies to implement.

3. **The Deploy Phase**

Now it's time to put it all into action. In the Deploy phase, this means putting into place our trunk, branching and merging strategy in a way that supports your product development.