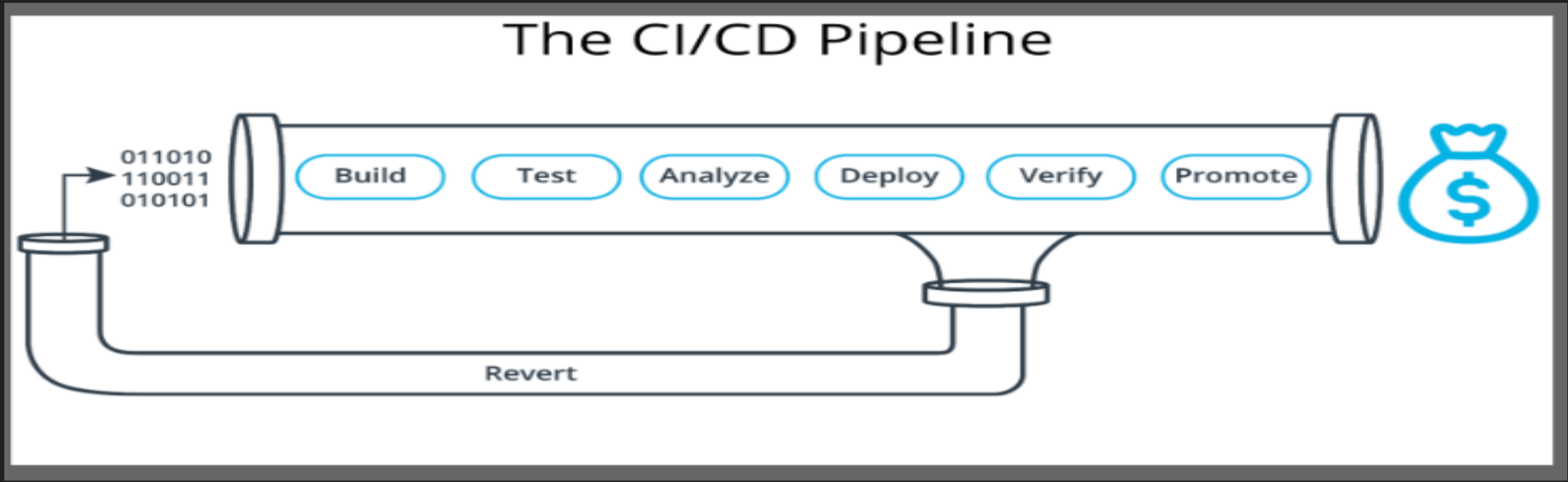
Fundamentals and Benefits of CI/CD to achieve, build, and deploy automation for cloud-based software products

Fundamentals of CI/CD

- One of the biggest hassles in software development is releasing a software which can also be time-consuming. Integration, configuration and testing must be done before a software can be released.
- Continuous integration, delivery and deployment comes in place to enable organizations make releases frequently through automated pipelines that handles all the repetitive tasks of building, testing and deployments.
- Continuous integration is the process or practice of merging all developers working copies to a shared main copy continuously.
- Continuous delivery – this is an engineering practice in which teams produce and release value regularly in short cycles.
- Continuous deployment is the approach of making sure all the working changes are delivered frequently through automated deployments.



- Pipeline is a process that facilitates the movement of software development through a specified path of building, testing and deploying code.
- The process involves feeding code into the pipeline which gets built, tested, analyzed, deployed, verified and promoted but in case of an issue in between the series a reversion happens and a stable running version of the code is retained.

Benefits of CI/CD

- Faster time to market (Increase revenue) – getting your product faster to the market is a big achievement to any organization and CI/CD can you make you achieve this. This is achieved because you can ship your products hourly, daily, weekly etc. thus new features can be released as soon as they are finished. This increases revenue because products ship faster to market.
- Better code quality (Avoid Cost)– testing is an essential step in software development which can also be time consuming. CI/CD provides automated tests functionalities that can run on every build. Automated tests are quicker to run thus can test more in a short time. Automated tests can also allow you discover bugs sooner as the tests run frequently as soon as a build is initiated, this enhances better code quality. This avoids costs as there will be less bugs in production and less time to do testing.
- Faster bug fixes (Reduce Cost)– Releasing changes regularly and in small bits will make it easier to identify an issue in case there is one because you will have a small bit of changes to work with. This reduces cost as developers will spend less time trying to figure out bugs

- Efficient infrastructure Creation (Avoid cost) – using infrastructure as code approach which involves automating creation of environments to run or serve the products. Having this as scripts ensures the environments are brought online quickly and consistently. This avoids cost by ensuring there are no human errors which can cause spinning up of wrong instances.
- Efficient Infrastructure cleanup (Reduce cost) – having scripts to handle infrastructure cleanup will go away to ensuring there is no additional costs raised by unused resources.
- Smooth Production path (Increases revenue)– following CI/CD practices will go a long way to ensuring your path to production to be always smooth. The steps involved will remain to be consistent. This increases revenue because revenue generating features can be released quickly and smoothly.
- Automated rollback (Protect revenue) – whenever there is an error or a bug when building CI/CD practices will do a rollback to a latest stable build thus ensuring all the operations are still running as expected as the issue gets sorted. This protects revenue as users will never get to interact with a product that has bugs.