

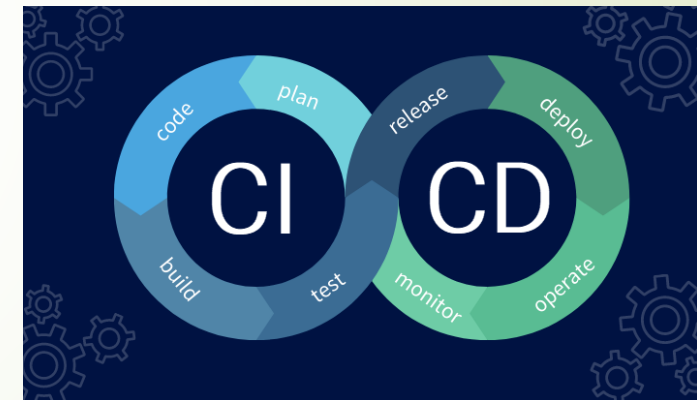
# CI/CD a top-notch solution for build/ship

- Demystifying CICD ?
  - When we think about a fully automated pipeline that is consistent, promising, and requires minimal human interference we do talk about CICD!

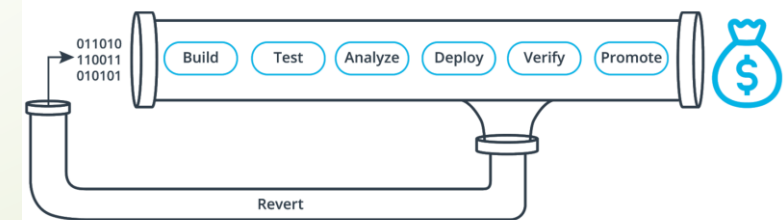
Has to balance bugs and new features

Has too many requirements

Wants to meet customer needs



The CI/CD Pipeline







# Continuous Integration

- CI is the act or practice of combining all developers' work into a unified master production-line frequently while maintaining a high quality, and bug-free artifact!
- A fully assembled solution that maintains the complete code development flow.
- If you had a look at the pipeline below, you would see how CI turns the hectic processes of delivering an artifact to a facile and simple one.
- Starting from the code source to storing a reliable build, CI provides an autonomous library of tools that our developers can use to facilitate and automate the whole workflow just by turning the long development cycle into a click of a button script

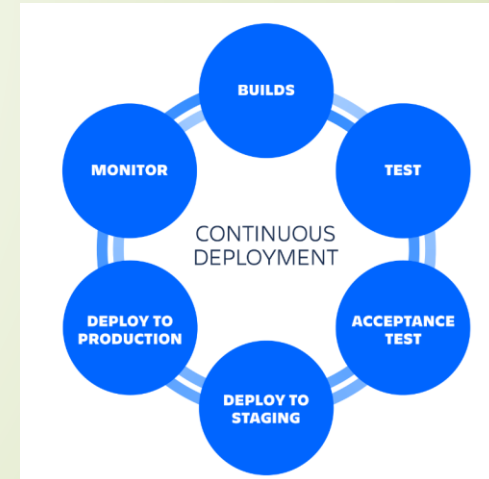
## Tool options for each stage of the CI/CD pipeline

A typical CI/CD pipeline has four stages, and each task can be addressed by multiple kinds of tools.

Source	Build	Test	Deliver/Deploy
<p><b>IDES AND CODE EDITORS</b> Atom, Cloud9, Microsoft Visual C++ or Visual Studio, PyCharm, Xcode</p> <p><b>VERSION CONTROL SYSTEMS</b> Git, Cloudsmith Package, Docker Hub (for container projects), JFrog Artifactory, nuGet, SVN</p>	<p><b>BUILD MANAGERS</b> Ant, Gradle, Make, Maven, Meister, Phing, Rake</p> <p><b>SECURITY TESTING TOOLS</b> Arctic Wolf Vulnerability Scan, Fortify Static Code Analyzer, Netsparker</p>	<p>Katalon, Kobiton, Kualitee, MicroFocus, QAProSoft, SauceLabs, Selenium, Telerik Test Studio, TestArchitect</p>	<p>Ansible Tower, Atlassian Bamboo, AWS CodeDeploy, Chef, CircleCI, Codeship, Deploybot, ElectricFlow, Jenkins, Octopus Deploy, Spinnaker, TeamCity</p>
			

# Continuous Deployment

- CD is rather an approach in which a product is delivered frequently through an autonomous flow. i.e: It's the process of releasing our in-house software to spotlight with an uninterrupted manner.
- Instead of spending much of working hours and employees' power in integrating their builds, testing them, provisioning, and maintaining an Infrastructure and finally deploying, CD has come with a multi-stage flow that ease the whole process and makes a developer's role just focused on innovation and less redundant
- If we look at CD in a brief overview, we will notice how are the different deployment stages we do regularly do separately connected and integrated, starting from creating the infrastructure heading to provisioning the servers, migrating the databases, and verifying they work as expected, without forgetting about rollbacks and promotions that CD detects and handles automatically!



# How it would payback!

- ▶ Starting to think of what do we really want from our business in terms of value, is what will really establish a solid top-notch product.
  - CI/CD will eventually create a topline deliverable release cycles with minimal cost and maximum revenue.
  - Reflecting on what's mentioned in the previous slide, we can map technical benefits CI/CD would bring to a tangible value.
  - Starting from the CI phase, we can now see how it would directly reflect on less bugs and less developer working hours that will be saved in automating catching compilation and unit test failures that will translate directly to a more firm, reliable software that ultimately goals to a bug-free artifact!
  - The tests CI supplies would prevent security holes that sometimes leaks in our releases.
  - In addition to that, CI promises a faster deployment scheme, since the process of Infrastructure creation and cleanup is now automated and monitored, that will be observed in our budget soon after inheriting the concept as we will be able to avoid idle resources that has high running cost without actually being used!



- To have the full profile of Continuous delivery, we should also turn an eye to what CD would bring to us!
  - After having a complete overview to the development and integration cycle, an autonomous deployment flow is now the key to a successful profile.
  - CD has proven a success story when digested since it will facilitate the production deployments that will instantly reflect on generating new value-generating features in a less time window than our customer's expectations.
  - One more aspect that we should never forget about is automating the manual Deploy-to-production checks and how we will have less time to market releases.
  - Protecting our revenue is one of the most important traits CD would take care of, since it will lessen the downtime from a major bug, and even help anticipating potential ones.
  - At last, Job failures and guilty patches are what really affect a software's house reputation and accordingly revenue turbulence. CD didn't drop this out since it will quickly undo to return to the original working state thanks to its automated rollback method that gets automatically triggered by any detected job failure!