Continuous

Integration
Deployment
Delivery

Powered by <epam>

CLOUD & DEVOPS COLOMBIA

CONTENT

- Start from the beginning: SDLC
- What is Continuous Integration?
- What is Continuous Delivery?
- What is Continuous Deployment?
- What is a CI/CD pipeline?
- Generic Pipeline steps
- CI/CD tools and engines

CLOUD & DEVOPS COLOMBIA

Ivan Gonzalez

Systems Engineer | DevOps Engineer

With 4+ years of experience in DevOps, I manage to get enough experience to share with people that are just beginning their journey into DevOps and its world.

Start from the beginning: SDLC

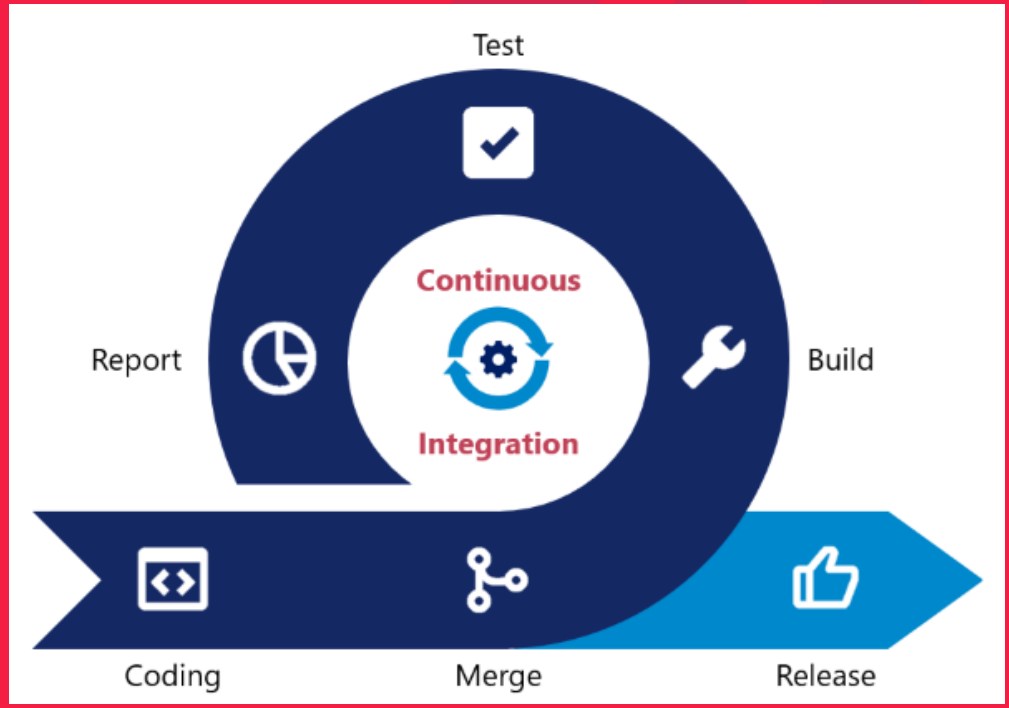
- Software Development Life Cycle (SDLC) is how the software is developed.
- There are some “frameworks” (or methodologies), that make flows and determinate what artifacts to deliver, how and where. Some of the examples are: Agile, Lean, Waterfall, etc.
- Depending on the methodology, the development is faster, cleaner and transparent are for all the stakeholders.
- DevOps is a “methodology” we can apply to SDLC, but CI/CD practice with its technologies impact the most into software development life cycle.

What is Continuous Integration?

Is the automated processes to integrate the code from the developers into a development branch.

Some of its generic components are:

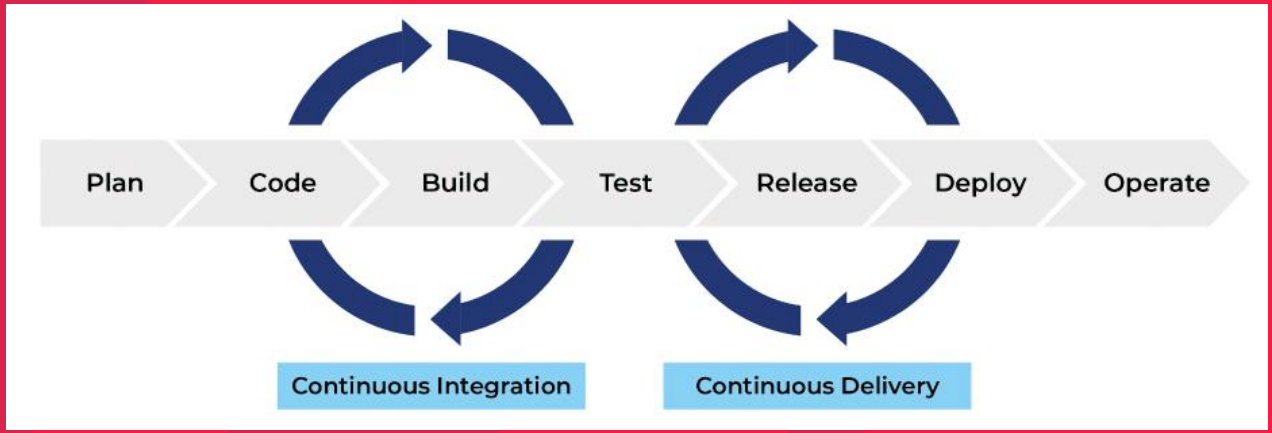
- Code: More than coding, is the part where the commit and push is done by the developers.
- Merge: Merge the feature branch against the development branch, to integrate the code.
- Build: Generate the artifacts that going to be deployed (compile the code, build the docker image, etc.).
- Testing: Run unit testing (testing over the methods of the code without a running service.).
- Report: Run a test coverage with a static code analysis tool to check the cleanness and security of the code, allowing the build being secure and ready to deploy.



What is Continuous Delivery?

Is the automated processes to deploy applications and services into non-productive environments. If we deploy into production in a manually, it would still be Continuous Delivery, due to it being not automated.

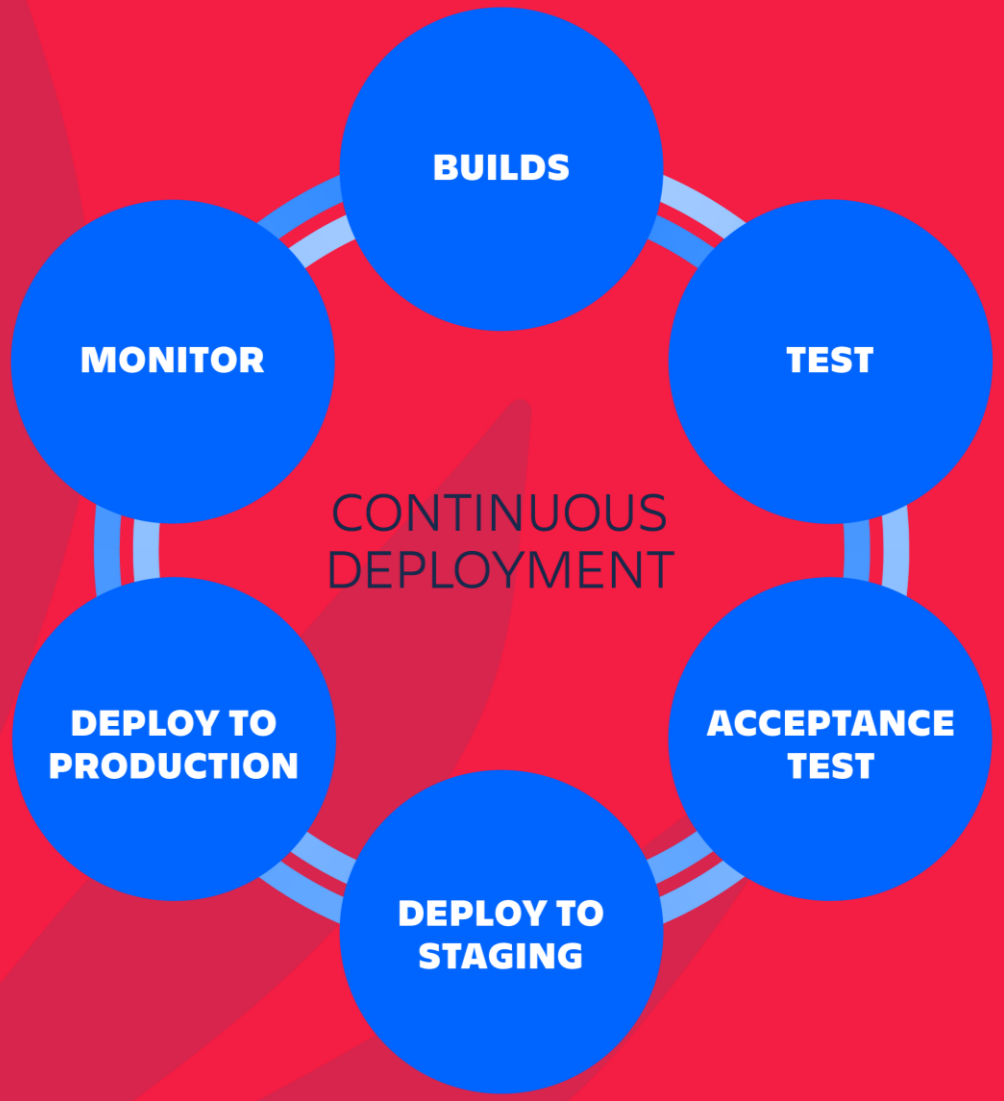
Continuous Delivery contains Continuous Integration, that why we can look at the CD as a cycle, due to the feedback this process gives to the stakeholders.



Testing against QA and DEV environments: After CI is done, testing in non-productive environments is done to ensure devs test their features, and QA teams does its manual testing to the features. CD ensure that there must be automated integration, API and GUI testing using the right tools and framework to do it.

Deploy to UAT (User Acceptance Testing) environments: this ensures that the services can be test in homogeneous environments (same as prod) and have approvals from stakeholders. This is the manual step that differentiates Delivery from Deployment.

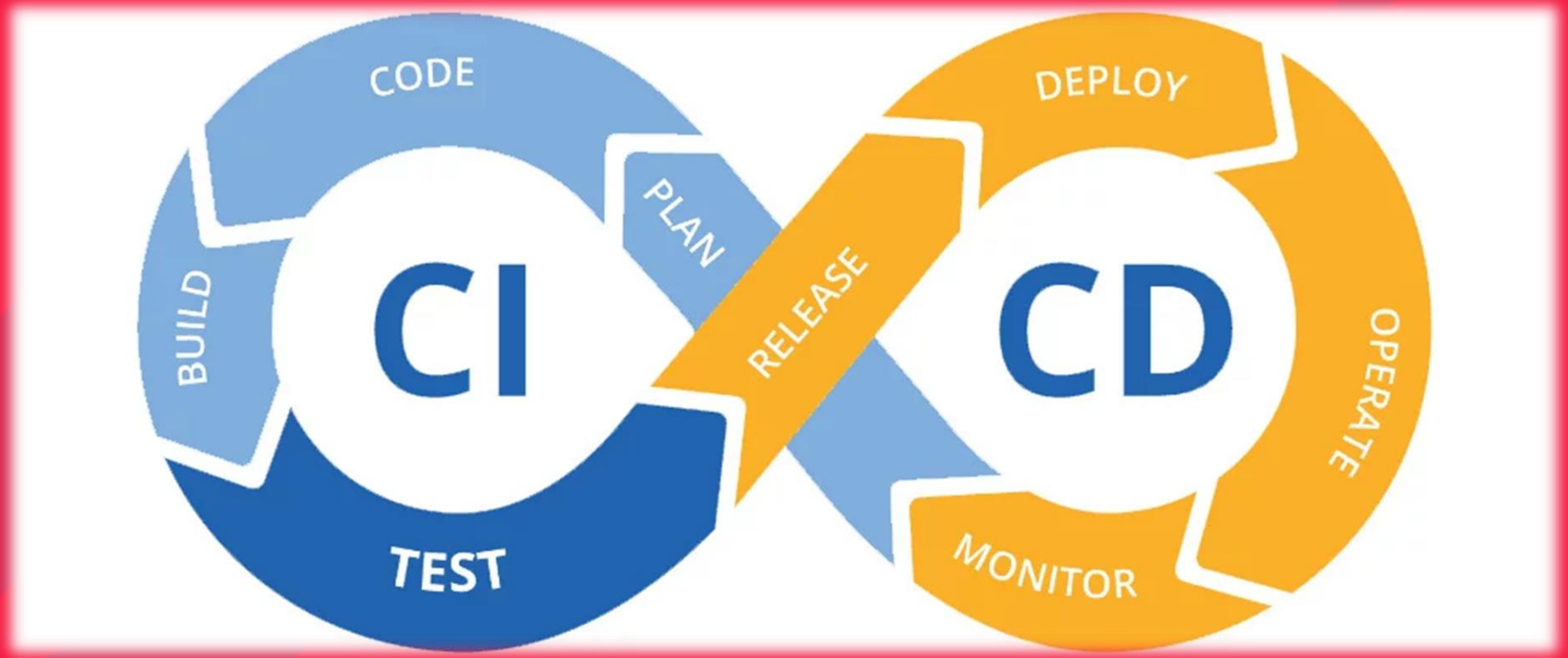
What is Continuous Deployment?



- Is the automated processes to deploy an application or service into production without manual approvals, all in one flow.
- The deployment must be automated, and all UAT must be automated against to its environment, ensuring that the applications or services are ready.
- But this flow doesn’t stop here, because we need to provide feedback in every step of the flow.
- Embedded is Observability and Monitoring, allowing to get metrics and information from everything, sharping decision making for new features, detecting bugs and security vulnerabilities before they affects the product.

What is a CI/CD pipeline?

The pipeline is the way we can develop the processes to accomplish the automation and achieve faster deployments and even faster feedback.



Generic Pipeline steps

Get the code: Clone the repository. In most of the CI/CD tools, we checkout the code from the repo.

Build: Compile the code, package it, build an image, etc. In this step we build the artifacts we going to deploy later.

Unit test: Run the unit tests against the code.

Static code analysis: Using a Static analyzer to get test coverage and bugs and security vulnerabilities.

Deploy to dev environment: Usually, the pipelines deploy the artifacts into this lower environments, for developers to test their features.

Deploy to QA environments: Simultaneously, the pipeline will deploy the artifacts into those environments, allowing QA teams to test faster, and the automated tests run on those environments.

Deploy to UAT environments: After QA testing is over and approved, deploy and testing in UAT is made to ensures the code is ready for production.

Deploy to Production environments: Deploy to production! That's the goal but not the end.

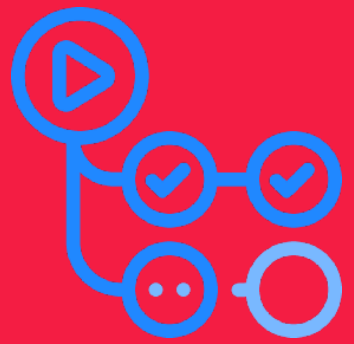
Monitoring: Even though we have monitoring in every step, monitoring after the deploying to production is fundamental to detect bugs and security vulnerabilities. Also, provide all the feedback ready for new features.

CI/CD tools and engines

Jenkins: Versatile, works for almost every project. Open source based in java and use groovy and its declarative pipeline language.

Azure DevOps: PaaS and SaaS, a ready to go solution for cloud but also (with some work) for on-premise environments. Use YAML as the main language.

Github Actions: SaaS embedded on Github, ready to build CI/CD pipelines in YAML.



On hands Demo Time!

THANK YOU!

Do you have any question?

Ivan_Gonzalez@epam.com