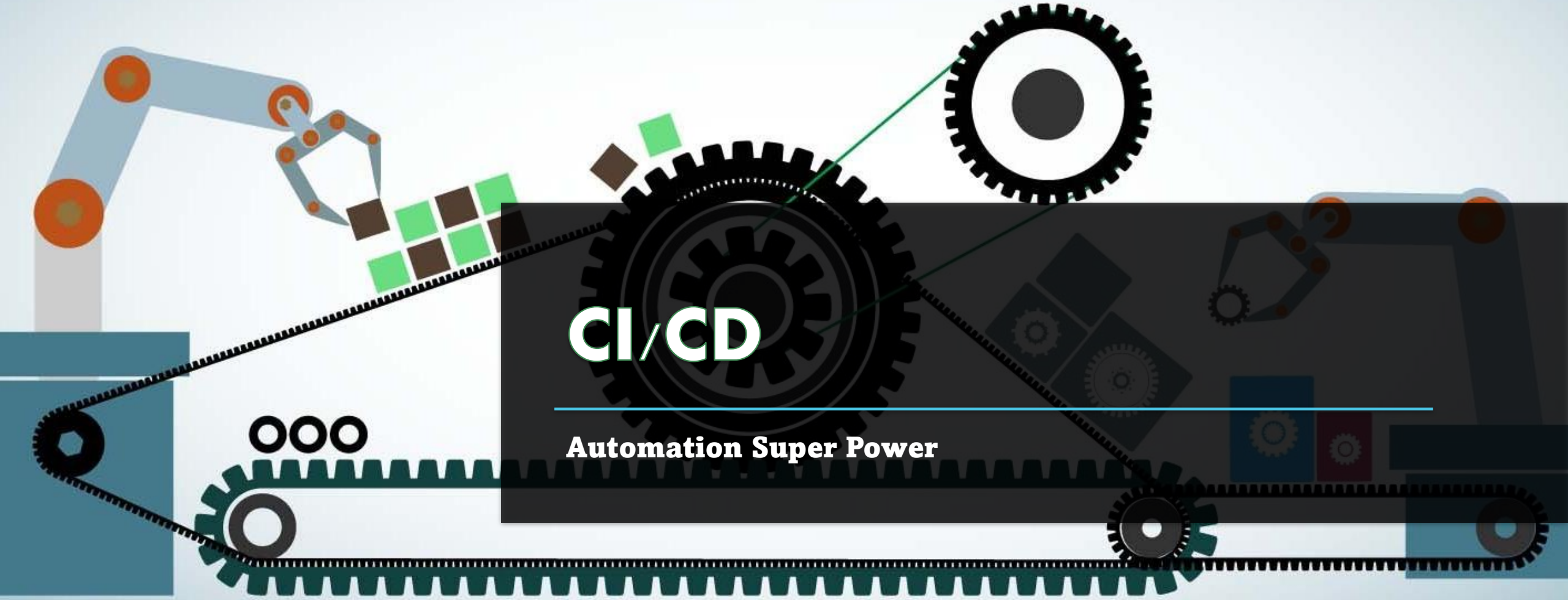


CI/CD

Automation Super Power

Representer By: Mohamed Emad El-Din

WHAT IS CI/CD ?

A continuous integration and continuous deployment (CI/CD) pipeline is a series of steps that must be performed in order to deliver a new version of software. CI/CD pipelines are a practice focused on improving software delivery throughout the software development life cycle via automation.

By automating CI/CD throughout development, testing, production, and monitoring phases of the software development lifecycle, organizations are able to develop higher quality code, faster. Although it's possible to manually execute each of the steps of a CI/CD pipeline, the true value of CI/CD pipelines is realized through automation.

Representer By: Mohamed Emad El-Din

CONTIUS INTEGRATION CI

is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates

Representer By: Mohamed Emad El-Din

CONTINUOUS DELIVERY, DEPLOYMENT CD

is a software development practice where code changes are automatically prepared for a release to production. A pillar of modern application development, continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When properly implemented, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

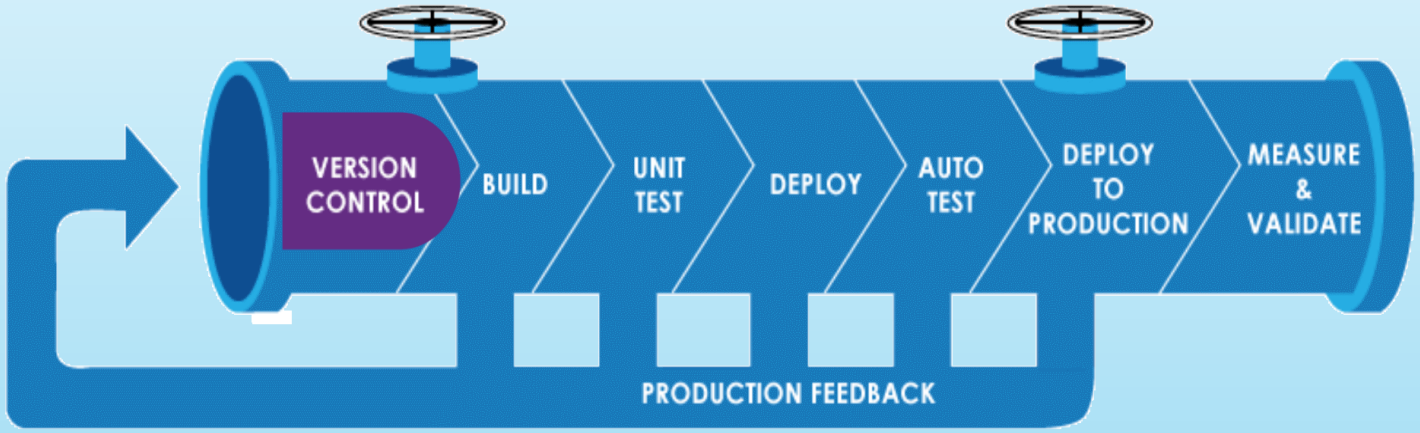
Representer By: Mohamed Emad El-Din

WHAT'S A CI/CD PIPELINE?

A pipeline is a process that drives software development through a path of building, testing, and deploying code, also known as CI/CD. By automating the process, the objective is to minimize human error and maintain a consistent process for how software is released. Tools that are included in the pipeline could include compiling code, unit tests, code analysis, security, and binaries creation.

Representer By: Mohamed Emad El-Din

In short, CI is a set of practices performed *as developers are writing* code, and CD is a set of practices performed *after* the code is completed.



Elements or stages in CI CD pipeline

Just refer the above diagram and you will get a brief overview of how the CI CD pipeline works. The pipeline is a logical demonstration of how software will move along the various phases or stages in this lifecycle before it is delivered to the customer or before it is live on production.

Representer By: Mohamed Emad El-Din

WHY CI IS SO GOOD ?

- Decoupled stages

Each step in CI Should do a single focused task

- Repeatable

Automated in a way that is consistently repeatable
Tooling should work for local developers too – local/remote parity

- Fail fast

Fail at the first sign of trouble

WHY CD IS SO GOOD ?

- Design with system in mind

Cover as many parts of a deployment as possible.
Application | Infrastructure | Configuration | Data

- Pipelines

Continually increase confidences as you move towards production

- Globally unique versions

Know the state of the system at anytime.
Be able to demonstrate difference between current and future state.

Continuous Integration
Continuous deployment
CI/CD Benfits

Representer By: Mohamed Emad El-Din

- Reduction of delivery risk

✓ No longer do we need to rely on humans with specific knowledge as the gate keepers of quality.
✓ Reduce chance of humans not following the process.
✓ Reduce chance of mis-communication on executing the change.

- To encode the process, we need to know the process

✓ If we know the tests pass
✓ If we know all the steps in deployment
✓ What is stopping us for releasing?

- Better visibility on change

✓ As our systems and tools are version controlled.
✓ And we know what the current state of production is
✓ And we can describe the process states with confidence

- Opens more avenues for review and increased audit compliance

- Increase efficiency and delivery options

✓ Enable us to deliver things with reduced effort.
✓ This leads us to deploy change more frequently.
✓ Which leads to getting feedback faster.
✓ That enables us to experiment easier.
✓ This leads smaller batch sizes.
✓ Which leads to and increase flow of the entire system.

- Enhance learning from failure

✓ When we have an issue or failure, we write a test to cover it.
✓ This test gets added to our suite an executed every time.
✓ Decreases our risk of this issue occurring again