# Continuous Delivery

Udacity Cloud DevOps Engineer Nanodegree - Project 3

Nawfal Tachfine

## What is Continuous Delivery?

- A workflow paradigm based on producing and delivering **value** in short **cycles**.
- In the context of software engineering, it is achieved by combining continuous **integration** and **delivery.**

**Importance**

- Gets user feedback at a higher rate to allow for actual **agility**.
- Improve **visibility** for business stakeholders and thus builds mutual **confidence**.

**Methodology**

- An **ideal** to constantly reach for by adding incremental improvements to workflow.
- **Adaptation** and clear **communication** is necessary for the entire organization
  - Feature requirements and characteristics are well defined.
  - Features are sliced into smallest valuable increments.
  - Delivery is the common goal to which all parties involved (both business and technical) are held accountable.

## Principles of Continuous Delivery

1. Releasing and/or deploying software follows **repeatable** and **reliable** processes.
2. These processes are entirely **automated**.
3. Everything is **version-controlled**.
4. Problematic steps of the process are brought forward for investigation and amelioration.
5. **Quality** is expected, measured, and improves with time.
6. A feature is said to be “done” when it has been **released** to the end-users.
7. Responsibility for delivery is shared between everyone involved.
8. Intentional and incremental steps are **continuously** taken for global **improvement**.

See “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation” by Humble & Farley.

## Benefits of Continuous Delivery

- Less bugs in production and less security vulnerabilities.
- Developers can focus more on producing code and less on peripheral low-yield tasks.
- Quick and easy rollbacks leading to less downtime if case of an incident.
- Less opportunities for human error in infrastructure management, making for faster and smoother deployments.
- Costs from unused infrastructure are eliminated.
- New value-generating features reach the users more quickly, reducing time to market and increasing revenue.

## Continuous Integration

CI is the practice of merging all working copies of a software product into a mainline as soon as possible. A CI pipeline performs some or all of the following steps:

- Compilation.
- Unit tests.
- Static code analysis.
- Dependency vulnerability testing.
- Storing the resulting artifacts.

## Continuous Deployment

CD is the practice of releasing software features to end-users as soon as they are ready, i.e whenever the codebase mainline gets updated. A CD pipeline performs some or all of the following steps:

- Creating infrastructure
- Provisioning servers
- Copying files and/or artifacts
- Promoting to production (blue/green, canary, A/B testing)
- Verification a.k.a smoke testing
- Rollbacking changes if any CD step fails