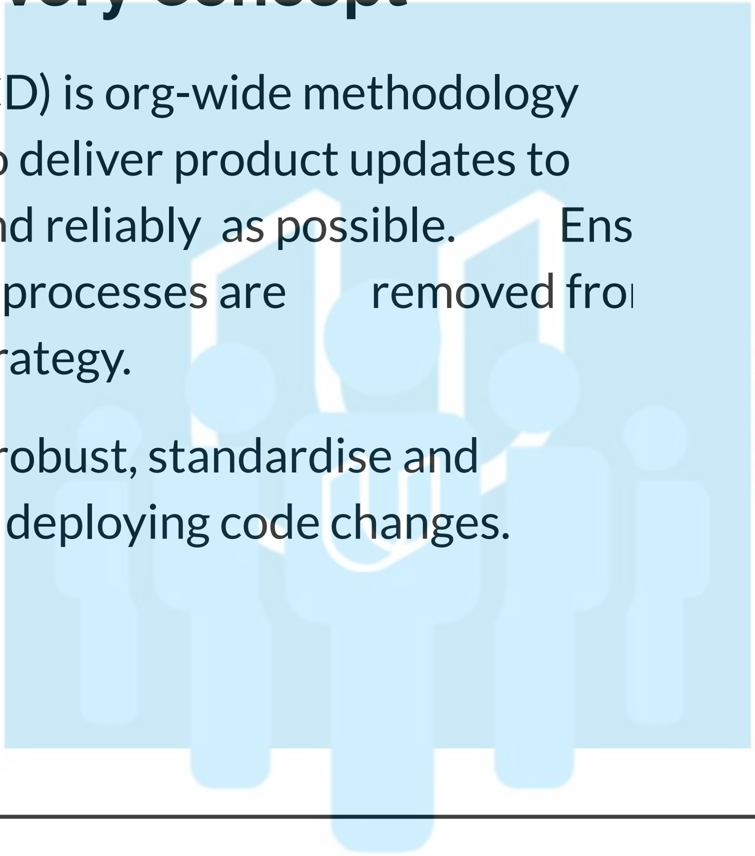
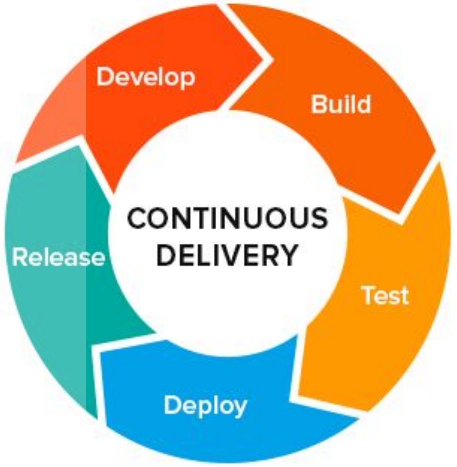
Continuous Delivery

Continuous Delivery Concept

Continuous Delivery (CD) is org-wide methodology defined as the ability to deliver product updates to customers as quickly and reliably as possible. Ens where possible manual processes are removed fro pipeline deployment strategy.

The aim is to provide a robust, standardise and automated process for deploying code changes.





8 Principles of Continuous Delivery (1 / 2)

1. **Processes are repeatable reliable**
   Automating as many tasks as possible provides predictability to each stage of the delivery process. This therefore cuts down on time wasted on troubleshooting within the deployment pipeline.
2. **Automate Everything**
   If it can be automated it should be. This cuts down on manual steps which then can lead to unforced errors in your processes. Once steps are automated they help build confidence in your repeatable reliable processes.
3. **Version Control**
   Everything means everything. Code, documentation, tests
4. **Bring the Pain Forward**
   Tackling painful or difficult tasks head on instead of ignoring it and pushing it back to deal with at a later release cycle.

8 Principles of Continuous Delivery (2 / 2)

5. **Build-in Quality**
   Building in checks and metrics to ensure quality is maintained or improved. Create baseline expectations for systems and use statistics to maintain standards. If you can measure your environment you can improve it!

6. **"Done" means released**
   A mindset across the board stating a process is not 'done' until it is successfully released to production. This means nobody has a half way when it comes to delivery.

7. **Everyone is Responsible**
   Shares the responsibility between all staff involved in the deployment cycle. This means there should be no siloed workloads, nobody should be able to hand over unfinished work. The team as a whole work as one to deliver a robust product.

8. **Continuous Improvement**
   Be comfortable and happy with small daily improvements

Benefits to Continuous Delivery Methodology

Reduce Costs

- Less bugs making their way into production and less time testing.
- Rapid regress and return to previous production environment.
- Faster and more frequent deployments using automation.
- Identify costly security holes within the code.
- Reduce infrastructure costs as resources are only spun up for the minimum amount of time possible.

Increase Productivity

- Identification of deployment issues earlier in the pipeline therefore reducing lost developer time.
- Rapid regress and return to previous production environment.
- Reduced ‘time to market’ for app improvements.
- Minimise downtime from bugs or errors found in the newly deployed code.