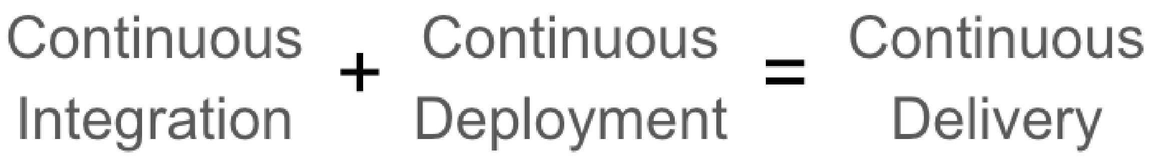
Introduction to Continuous Delivery

There is some confusion in the community around Continuous Delivery, Continuous Integration and Continuous Deployment.



The relationship between continuous integration, delivery, and deployment depicted.

Continuous Delivery Is Important

Devops, Agile, Lean, Kanban, Scrum... all great things, all exciting buzz words, each the subject of many conferences, blogs. But, not a single one of those amazing concepts or methodologies will make a real difference until we allow them to change how we think about the value we deliver to our customers.

Continuous Delivery doesn't replace anything, but rather it enhances everything.

Use CD to Add Value

Both sides of the business/tech divide need to involve the other side in decisions so that our companies can be stronger and make a better impact on the world. But, since, **8 Principles of Continuous Delivery**

1. Repeatable Reliable Process
2. Automate Everything
3. Version Control Everything
4. Bring the Pain Forward
5. Build-in Quality
6. "Done" Means Released
7. Everyone is Responsible
8. Continuous Improvement

Where Does CI/CD Fit In?

Stage	***Before* CI/CD**	***After* CI/CD**
Coding	*Human*	*Human*
Code Review	*Human*, Subjective, Inconsistent	*Human*/CI - Static Analysis
Compile/Lint	*Human*	CI
Merge/Integrate	*Human*	CI
Run Unit Tests	*Human*, Hit or Miss, Easily Bought Off with Pressure	CI
Run Integration Tests	*Human*, Hit or Miss, Easily Bought Off with Pressure	CI
Verify Dependency Security	*Human*, Often Not Done	CI
Deploy to Test Env	*Human*, Problematic, Missed Steps	CD

Stage	***Before* CI/CD**	***After* CI/CD**
Team Test	*Human*, Time Consuming	CD - Automated Acceptance Tests
Deploy to Client Test Env	*Human*, Problematic, Missed Steps	CD
Client Test	*Human*, Often Unnecessary If Pre-Development Activities are On Point	*Human* - Maybe Not Needed If We Can Build Confidence
Create Infrastructure	*Human*, Problematic, Missed Steps, Stressful	CD
Deploy to Production	*Human*, Problematic, Missed Steps, Stressful	CD
Smoke Test in Prod	*Human*, Inconsistent	Automated Smoke Tests (Subset of AAT's)
Rollbacks	*Human*, Problematic, Missed Steps, Stressful	CD
Promoting Production	*Human*, Problematic, Missed Steps, Stressful	CD
Celebrate!	*Human*	*Human*

Removing *Human* Error

Before we implement CI/CD almost *everything* requires human intervention. Can you imagine a world without human error?
... Neither can I, but with CI/CD, we can reduce it!

How do you know you need CI/CD or Continuous Delivery?

There are several "warning signs" that teams exhibit that suggest they would be good candidates for CI/CD or Continuous Delivery. If you identify with any of these items, you should consider CI/CD an essential piece of your development workflow.

- Investing **more time** in a release cycle than delivering value
- Going through integration hell every time we finish a feature
- **Code gets lost** because of botched merges
- Unit test suite hasn't been green in ages
- Deployments contribute to **schedule slip**
- Friction between ops and development departments
- **Only one engineer** can deploy a system
- ***Deployments are not cause for celebration***

No Free Lunch

No pain, no gain, right? Did you think CI/CD was going to solve all your woes and ask nothing in return? Think again!

- No more manual deploying to environments
- No more modifying environment settings in GUI's
- No more neglecting the unit tests
- No more leaving broken code in place
- Requires a high level of discipline
- Requires additional skills to maintain and extend automation

Here I can Explain some Benefits of CI/CD in our company.

Technical Language	Value	Translation
Catch Compile Errors After Merge	Reduce Cost	Less developer time on issues from new developer code
Catch Unit Test Failures	Avoid Cost	Less bugs in production and less time in testing
Detect Security Vulnerabilities	Avoid Cost	Prevent embarrassing or costly security holes
Automate Infrastructure Creation	Avoid Cost	Less human error, Faster deployments
Automate Infrastructure Cleanup	Reduce Cost	Less infrastructure costs from unused resources
Faster and More Frequent Production Deployments	Increase Revenue	New value-generating features released more quickly
Deploy to Production Without Manual Checks	Increase Revenue	Less time to market
Automated Smoke Tests	Protect Revenue	Reduced downtime from a deploy-related crash or major bug
Automated Rollback Triggered by Job Failure	Protect Revenue	Quick undo to return production to working state