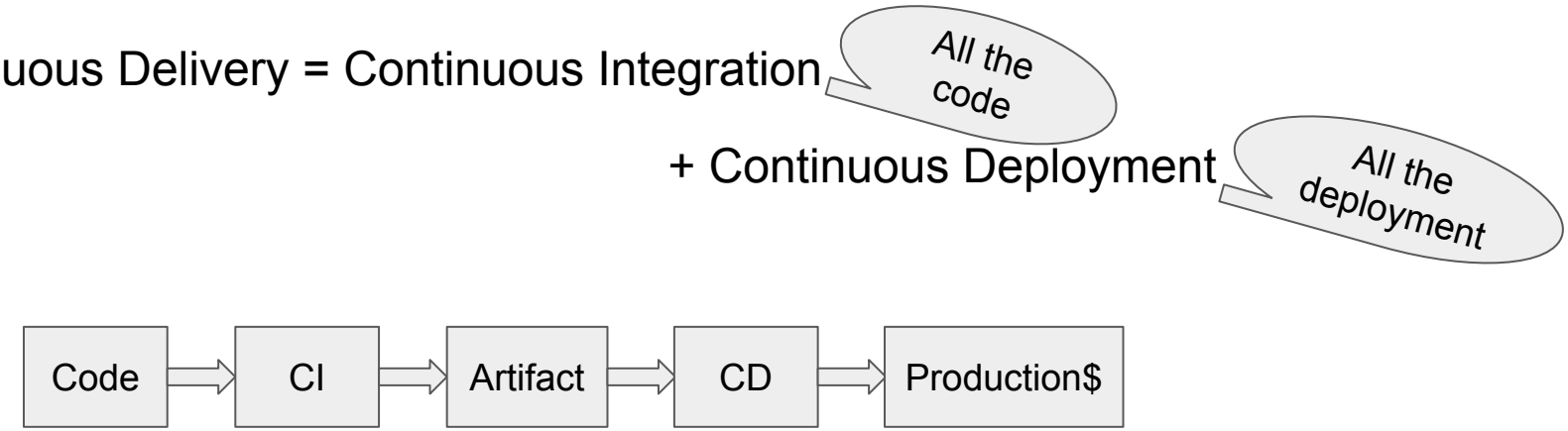
Why CI/CD?

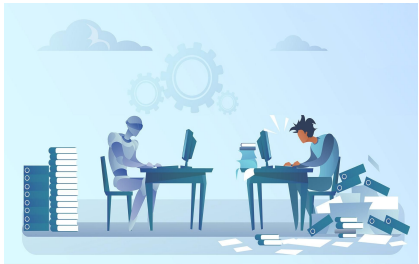
Development >< Operations!

Set, automate, gain

- Continuous Delivery = Continuous Integration + Continuous Deployment



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



CI/CD Value

Pursuit	**Value**	**Translation**
Kind of metric.	Protect cost.	Continuous enhancements and more retained subscriptions.
Catch coding errors.	Reduce cost.	Less developer time.
Catch unit test failures.	Avoid cost.	Less bugs in production.
Detect security vulnerabilities.	Avoid cost.	More robust product and less costs on maintaining it.
Automate infrastructure creation.	Avoid cost.	Less human errors, faster deployments.

CI/CD Value Continued

Pursuit	Value	Translation
Automate infrastructure cleanup.	Reduce cost.	Less unused resources costs.
More frequent production deployments.	Increase revenue.	Taking feedbacks of the customers into action immediately releasing new features more quickly.
Faster production deployments.	Increase revenue.	Less time to market.
Automated smoke tests.	Protect Revenue.	Reduced downtime from a deploy-related crash or major bug.

SOS, we need CI/CD

- Investing more time in a release cycle than delivering value.
- Going through integration pain every time finishing a feature.
- Testing practices skip because of being tight in time causing catastrophic production deployments.
- Deployments contribute to schedule slip.
- Devs and Ops departments are not allies.

Principles of Continuous Delivery

- Repeatable reliable process.
- Automate everything for less error-prone tasks.
- Version control everything for convenient roll-backs.
- Fail fast and loud.
- Build-in quality.
- "Done" means released with all the team involved, everyone is responsible.
- If you broke it, you fix it.
- Continuous improvement.

Any Questions?

Thank you!