Continuous Deployment

Continuous deployment (CD) benefits your software teams and customers. Learn what it is, the benefits, best practices, and more.

BY STEN PITTET

What is Continuous Deployment?

Continuous Deployment (CD) is a software release process that uses automated testing to validate if changes to a codebase are correct and stable for immediate autonomous deployment to a production environment.

The software release cycle has evolved over time. The legacy process of moving code from one machine to another and checking if it works as expected used to be an error prone and resource-heavy process. Now, tools can automate this entire deployment process, which allow engineering organizations to focus on core business needs instead of infrastructure overhead.

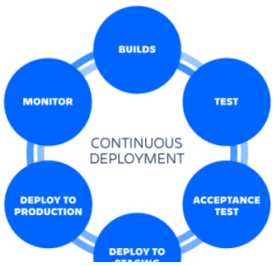
Continuous deployment vs. continuous delivery

The distinction between continuous deployment vs. continuous delivery can be confusing because of the nomenclature. They are both abbreviated as CD and have very similar responsibilities. Delivery is the precursor to deployment. In delivery, there is a final manual approval step before production release.

The following is a mnemonic exercise to help remember the distinction between the two. Think about receiving a package from your favorite online retail store. When waiting for a package to arrive you coordinate with a **delivery** service. This is the delivery phase. Once the package has successfully arrived, you open the package and review its contents to make sure it matches expectations. If it does not, it may be rejected and returned. If the package is correct you are ready to **deploy** and use the new purchase!

In the delivery phase, developers will review and merge code changes that are then packaged into an artifact. This package is then moved to a production environment where it awaits approval to be opened for deployment. In the deployment phase, the package is opened and reviewed with a system of automated checks. If the checks fail the package is rejected.

When the checks pass the package is automatically deployed to production. Continuous deployment is the full end to end, automated software deployment pipeline.



Advantages and disadvantages of continuous deployment

Continuous deployment offers incredible productivity benefits for modern software businesses. It allows businesses to respond to changing market demands and teams to rapidly deploy and validate new ideas and features.

With a continuous deployment pipeline in place, teams can react to customer feedback in real time. If customers submit bug reports or requests for features, teams can immediately respond and deploy responses. If the team has an idea for a new product or feature, it can be in the hands of customers as soon as the code has been pushed.

The benefits of continuous deployment, however, come at a price. While the return on investment is high, a continuous deployment pipeline can be an expensive initial engineering cost. In addition to the initial cost, on going engineering maintenance may be required to ensure the pipeline continues to run fast and smooth.

Continuous Deployment Tools

Establishing continuous deployment requires substantial engineering investment. The following is a list of tools that are needed to build a continuous deployment pipeline.

- Automated testing
  The most critical dependency for continuous deployment is automated testing. In fact, the entire chain of continuous integration, delivery and deployment depends on it. Automated tests are used to prevent any regressions when new code is introduced and can replace manual reviews of new code changes.
- Rolling deployments
  The distinguishing feature between continuous deployment and delivery is the automated step of activating new code within a live environment. A continuous deployment pipeline must be able to undo a deployment in the event that bugs or breaking changes are deployed. Automated rolling deployment tools like green-blue deploys are a requirement for proper continuous deployment.
- Monitoring and alerts
  A robust continuous deployment pipeline will have real time monitoring and alerts. These tools provide visibility into the health of the overall system and into the before and after state of new code deployments. Additionally alerts can be used to trigger a rolling deployment 'undo' to revert a failed deploy.

Continuous Deployment Best Practices

Once a continuous deployment pipeline is established, ongoing maintenance and participation is required from the engineering team to ensure its success. The following best practices and behaviors will ensure an engineering team is getting the most value out of a continuous deployment pipeline.

- Test-driven development
  Test-driven development is the practice of defining a behavior spec for new software features before development begins. Once the spec is defined developers will then write automated tests that match the spec. Finally, the actual deliverable code is written to satisfy the test cases and match the spec. This process ensures that all new code is covered with automated testing up front. The alternative to this is delivering the code first and then producing test coverage after. This leaves opportunity for gaps between the expected spec behavior and the produced code.
- Single method of deployment
  Once a continuous deployment pipeline is in place, it's critical that it is the only method of deployment. Developers should not be manually copying code to

production or live editing things. Manual changes external to the CD pipeline will desync the deployment history, breaking the CD flow.

- Containerization
  Containerizing a software application ensures that it behaves the same across any machine it is deployed on. This eliminates a whole class of issues where software works on one machine but behaves differently on another. Containers can be integrated as part of the CD pipeline so that the code behaves the same on a developer's machine as it does during automated testing, and production deployment.

Summary

Continuous deployment can be a powerful tool for modern engineering organizations. Deployment is the final step of the overall 'continuous pipeline' that consists of integration, delivery, and deployment. The true experience of continuous deployment is automation to the level at which code is deployed to production, tested for correctness, and automatically reverted when wrong, or accepted if correct.

STEN PITTET

I've been in the software business for 10 years now in various roles from development to product management. After spending the last 5 years in Atlassian working on Developer Tools I now write about building software. Outside of work I'm sharpening my fathering skills with a wonderful toddler.

ARTICLE

How to get to Continuous Deployment

Move from continuous integration to continuous deployment with these six strategies.

Read this article →

TUTORIAL

Continuous Deployment Tutorial

This tutorial will show you how to get started with continuous deployment with Bitbucket Pipelines.

Try this tutorial →

CI/CD Topics

Continuous Delivery
Continuous Integration
Continuous Deployment
Pipelines
Software Testing
Microservices
Tutorials

Up Next
How to get to Continuous Deployment →