

CONTINUOUS INTEGRATION ESSENTIALS

Learn all about Continuous Integration. If you're ready to get started, feel free to sign up for a free Codeship account!

SIGN UP FOR FREE

START A CONVERSATION

Codeship got ranked amongst the Top 5 CI services by Forrester.

[Download the Forrester Wave Report on Continuous Integration Tools here.](#)

In this guide you will learn about all things Continuous Integration, how it ties in with Continuous Deployment and Continuous Delivery and how to get started with these practices. Once you know about them we talk more in detail about best-practices and workflows and are providing a thorough list of resources at the end.

What is Continuous Integration?

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build

and automated tests. While automated testing is not strictly part of CI it is typically implied.

One of the key benefits of integrating regularly is that you can detect errors quickly and locate them more easily. As each change introduced is typically small, pinpointing the specific change that introduced a defect can be done quickly.

In recent years CI has become a best practice for software development and is guided by a set of key principles. Among them are revision control, build automation and automated testing.

Additionally, Continuous Deployment and Continuous Delivery have developed as best-practices for keeping your application deployable at any point or even pushing your main codebase automatically into production whenever new changes are brought into it. This allows your team to move fast while keeping high quality standards that can be checked automatically.

Continuous Integration doesn't get rid of bugs, but it does make them dramatically easier to find and remove.

[Martin Fowler](#), Chief Scientist, ThoughtWorks

What is the difference between Continuous Integration, Continuous Deployment & Continuous Delivery?

Continuous Integration

is the practice of integrating changes from different developers in the team into a mainline as early as possible, in best cases several times a day. This makes sure the code individual developers work on doesn't divert too much. When you combine the process with automated testing, continuous integration can enable your code to be dependable.

Continuous Deployment

is closely related to Continuous Integration and refers to keeping your application deployable at any point or even automatically releasing to a

test or production environment if the latest version passes all automated tests.

Continuous Delivery

is the practice of keeping your codebase deployable at any point. Beyond making sure your application passes automated tests it has to have all the configuration necessary to push it into production. Many teams then do push changes that pass the automated tests into a test or production environment immediately to ensure a fast development loop.

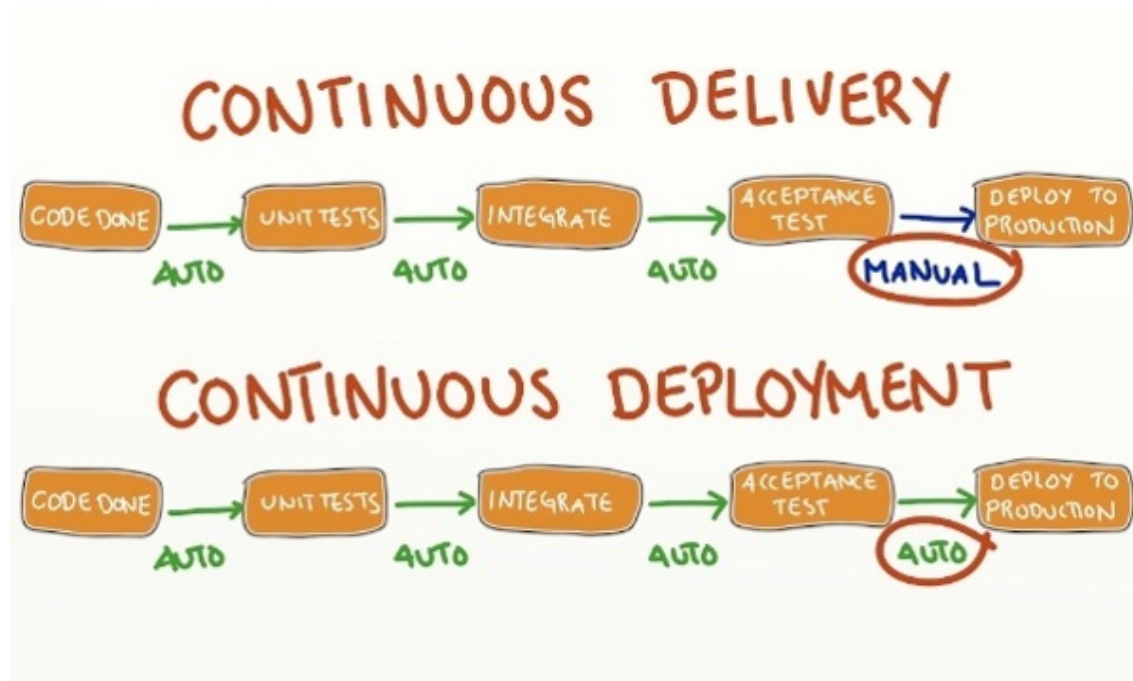


Image via Yassal Sundman ([Source](#)) Read more on the topic:

- ▶ [Continuous Delivery vs Continuous Deployment](#) by Jez Humble
- ▶ [Why You Should Use Continuous Integration and Continuous Deployment](#) by Florian Motlik
- ▶ [Continuous Delivery vs Continuous Deployment vs Continuous Integration - Wait huh?](#) by Michael Chletsos
- ▶ [Five Tips to Get Started with Continuous Delivery](#) by Florian Motlik
- ▶ [Continuous Deployment](#) by Timothy Fitz

- ▶ [Continuous Delivery](#) by Martin Fowler
- ▶ [Continuous Delivery](#) by Wikipedia
- ▶ [The Continuous Delivery Pipeline — What it is and Why it's so Important in Developing Software](#) by Andrew Phillips

Part 1: Beginners Guide to Continuous Integration

You should focus on setting up a simple **Continuous Integration** process as early as possible. But that's not where things should end. Even though Continuous Integration (CI) is important, it's only the first step in the process. You also want to set up Continuous Deployment (CD), the workflow that automates your software deployment and lets you focus on building your product.

Continuous Integration (CI) vs Continuous Deployment (CD)

As we pointed out before, Continuous Deployment is closely related to Continuous Integration and refers to keeping your application deployable at any point or even automatically releasing into production if the latest version passes all automated tests.

If you wish to [release your product really fast](#), you should automate your entire workflow, not just the testing. Having a well designed and smoothly running Continuous Deployment (CD) solution will be the glue between the tools you use, especially between the SCM (Source Control Management) provider/server and the hosting environment you are using. This will also help you to onboard new people and grow your team as they can rely on a fully automated process from day one.

Read more on the topic here: [Under the Hood at Thinkful: Continuous Integration \(CI\) Rollout](#) by Jason Blanchard

What is the best Continuous Integration & Deployment tool or service? How do I choose between them?

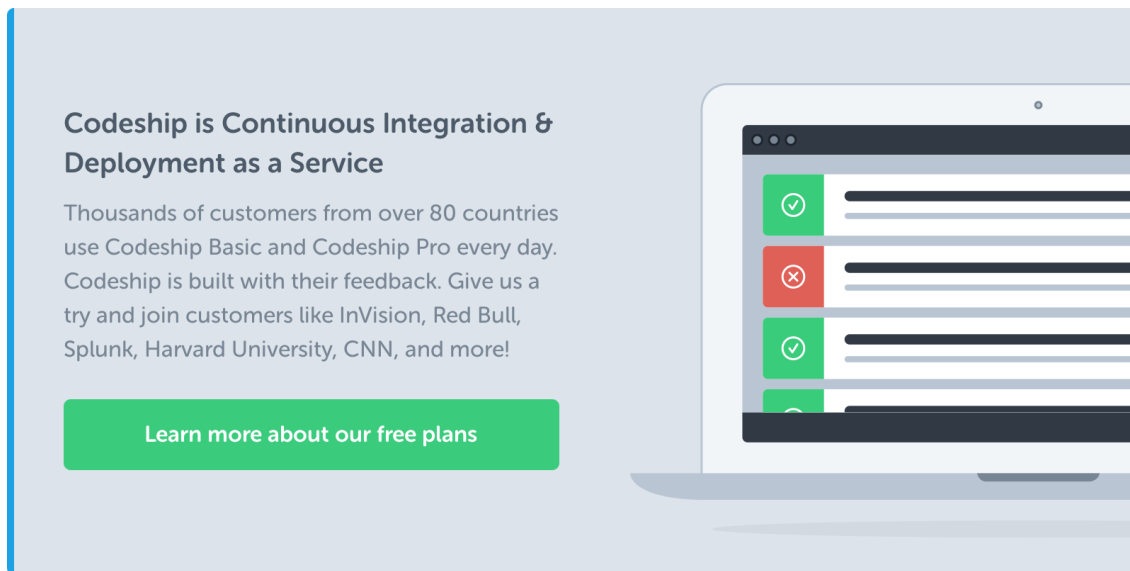
There are many solutions out there. If you only want to get a list of tools, you can look at Codeship, TravisCI, SemaphoreCI, CircleCI, Jenkins, Bamboo, Teamcity or many others. You can also find many articles and discussions out there on the topic with valuable information like [this one on Quora](#). There are almost endless opportunities out there. But then the question rises: *"How to choose between these?"*

Your pick will strongly depend:

- ▶ on your requirements,
- ▶ on the techstack you have and
- ▶ on how you handle your daily workflow.

As Codeship CEO Moritz Plassnig [points out on Quora](#), it usually helps to raise a couple of simple questions first and answer them before picking any solution. This will help you determine which solution would be the best fit for you.

- ▶ <http://stackshare.io/codeship>
- ▶ <https://www.g2crowd.com/products/codeship/reviews>
- ▶ <http://www.quora.com/Reviews-of-Codeship-Software/>



Codeship is Continuous Integration & Deployment as a Service

Thousands of customers from over 80 countries use Codeship Basic and Codeship Pro every day. Codeship is built with their feedback. Give us a try and join customers like InVision, Red Bull, Splunk, Harvard University, CNN, and more!

[Learn more about our free plans](#)

Hosted vs non-hosted solutions

One of the first decisions you have to make is whether you want a hosted Software-as-a-Service (SaaS) solution or a self-hosted solution.

If you prefer a self-hosted solution you need to administer your own server. The SaaS solution doesn't require this, but it might be more limiting in case you require some edge case features. If you happen to use GitHub, Bitbucket, Heroku, or other cloud services, then it is most likely that you want a SaaS solution as it will fit your already existing workflow.

If data security is very important, then a self-hosted server might be a better choice for you. SaaS solutions generally **let you focus more on your core product** as you don't have to spend time on maintaining your infrastructure and keeping all dependencies updated at the cost of some flexibility.

Testing Open Source vs proprietary software

If you have open source projects, you can test them with either solution. Be it a hosted one or a non-hosted one. Both of them have their pros and cons. As mentioned, a hosted (SaaS) solution doesn't require

maintenance of the servers on your side, which leaves more time for you to work / code on your product.

The vast majority of SaaS solutions follow the GitHub model and you can test your open source projects free of charge. Some open source projects do require a lot of control over the build infrastructure though as they might be testing parts of an operating system not accessible in a hosted solution. In this case any of the existing open source CI servers should do a good job, although with added necessary maintenance overhead.

Benefits and Advantages of Continuous Integration and Deployment

Continuous Integration has many benefits. A good CI setup speeds up your workflow and encourages the team to push every change without being afraid of breaking anything. There are more benefits to it than just working with a better software release process. Continuous Integration brings great business benefits as well.

Reduces Risk

If you test and deploy code more frequently, it will eventually reduce the risk level of the project you are working on as you can detect bugs and code defects earlier. This means they are easier to fix and you can fix them sooner which makes it cheaper to fix them. This will speed up the feedback mechanism and make your communication much smoother, as mentioned in this article by Intercom's Darragh Curran: [Shipping is your company's heartbeat](#).

Better Communication

When you have a CI process in place that is hooked into a Continuous Delivery workflow it's easy to share your code regularly. This code sharing helps to achieve more visibility and collaboration between team members. Eventually this increases communication speed and efficiency within your organization as everybody is on the same page, always.

Faster iterations

As you release code often, the gap between the application in production and the one the developer is working on will be much smaller. Your thinking about how to develop features most probably will change. As every small change will be tested automatically and the whole team can know about these changes you will want to work on small, incremental changes when developing new features. This results in less assumptions as you can build features quicker and test and deploy them automatically for your users to see as soon as possible, thus gaining valuable feedback from them faster.

Faster feedback on business decisions

Having a CI process is not only beneficial for software developers, but for their managers as well. Both parties can gather valuable feedback and gain insights much faster. As you push code more often, you have more data available which you can analyze to check if the product is heading into the right direction. This continuous data flow and the timeline of metrics (like dependency, [unit tests](#), complexity, and [code smell](#)) can also help to reflect on the progress of the project more frequently which enables faster technological and business decisions.

Some other benefits of using CI and CD

- ▶ Reduces overhead across the development and deployment process

- Reduces the time and effort for integrations of different code changes
- Enables a quick feedback mechanism on every change
- Allows earlier detection and prevention of defects
- Helps collaboration between team members so recent code is always shared
- Reduces manual testing effort
- Building features more incrementally saves time on the debugging side so you can focus on adding features
- First step into fully automating the whole release process
- Prevents divergence in different branches as they are integrated regularly
- If you have a long running feature you're working on, you can continuously integrate but hold back the release with [feature flags](#).

Read more on the topic here: [The Benefits of Continuous Integration](#) by Joe Green

Part 2: A deeper dive into Continuous Integration and Continuous Delivery

If you are interested in Continuous Integration tutorials and [best practices](#) we suggest you check out some of the engineering blogs mentioned below. You can find very helpful content there.

The CI and CD landscape is changing and shaping rapidly since 2006. Still, it's worth having a look though at Martin Fowler's original [principles of Continuous Integration](#). Martin explains the best practice workflow:

1. Maintain a code repository
2. Automate your build
3. Make your build self-testing

4. Daily commits to the baseline by everyone on the team
5. Every commit (to the baseline) should be built
6. Keep your builds fast
7. Clone the production environment and test there
8. Make it easy to get the latest deliverables
9. Everyone on the team can see the results of your latest build
10. Automate build deployment

The industry has been doing pretty well to enable this and software teams largely are able to work with these principles in mind. With the [emergence of containers](#) it's now a lot easier to clone your local and production environment and test there. Codeship's new Docker Platform will help you with exactly that and much more. Feel free to [learn more about it here](#).

Continuous Delivery checklist

Martin Fowler's principles are a great starting point to think about best setting up your software development process. Jez Humble and David Farley also point out in their book "[Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation](#)" that the following list should be a general outline and checklist when you want to submit code.

1. Before submitting changes, check to see if a build is currently in the "Successful" status. If not, you should assist in fixing a build before submitting new code.
2. If the status is currently "Successful", you should rebase your personal workspace to this configuration.
3. Build and test locally to ensure the update doesn't break functionality.
4. If Successful, check in new code.
5. Allow CI to complete with new changes.
6. If build **fails**, **stop** and **fix** on your machine. Return to step 3.

7. If build **passes**, **continue** to work on the next item.

You should note that this is only a general outline. You can find many services and solutions which do not follow these exact steps (like step 2). However, it is good to be aware of these steps. Why?

Because many developers (according to DZone's research in 2014 up to 41%) believe that they are achieving Continuous Delivery, while in fact less than 10% of them actually do. *(The survey was conducted on 500+ IT professionals. A large portion of the respondents were developers (68%) or team managers (14%) with smaller portions of operations, QA, and executive management represented. The majority of respondents were headquartered in the US (36%) or Europe (43%).)*

Less than 10% of these people actually work with Continuous Delivery. Let that sink in. This is one of the reasons why it is good to remind us to push ourselves to get closer to **real** Continuous Delivery. A good checklist definitely helps with setting up the right process and explaining it to your team and, potentially, management.

That's one of the reasons why DZone, the company behind the research, put together a checklist. In the **Continuous Delivery Maturity Checklist** you can actually check the practices you currently perform to see how mature you are in each area of Continuous Delivery. The higher you score on the test, the closer you are to achieving CD Maturity. The checklist is not only good to follow when you code, but it can also help you identify weaknesses and areas to improve in your company's CD process. The main areas of the CD process include:

- ▶ Source Control
- ▶ Build Process
- ▶ Testing & Q&A
- ▶ Deployment
- ▶ Visibility

You can download their [checklist here](#).

An earlier version of this process which you might want to have a look at was introduced by Chris Shayan, when he wrote about the [Continuous](#)

Delivery maturity matrix here.

Continuous Delivery maturity matrix

	Novice	Beginner	Intermediary	Advanced	Expert
Build	Verification before commit run in developer's Workspace Common nightly build	CI server builds on commit Artifacts are managed	No build scripts -only configurations Dependencies are managed	Distributed builds Staged build sequence	Build from VM CI server orchestrate VMs
Test + QA	Unit Test Code Coverage	Metrics on technical debt & compliance Mock-up's & proxies	Peer-reviews Automated Functional Test	Test Data Test in target	Automated Acceptance Test
SCM	"Early Branching" Branches used for releases Merges are rare	"Late branching" Branches used for work isolation Merges are common	Pre-tested Commits Integration branch is pristine	All commits are tied to tasks Individual history rewrites in DVCS	Release notes & traceability analysis are generated automatically
Visibility	Build status is notified to committer	Latest build status is available to all stakeholders	Trend reports Build status can be subscribed to (pull vs push)	Monitors in work areas show real-time status	Build reports and statistics are shared with customer and public

Source: <https://chrisshayan.atlassian.net/wiki/display/my/2013/07/23/Continuous+Delivery+Matrix>

Read more on the topic:

- [Continuous Delivery Maturity Model](#) by Eric Minick
- [Continuous Delivery Matrix](#) by Chris Shayan
- [Continuous Delivery Maturity Checklist](#) by DZone

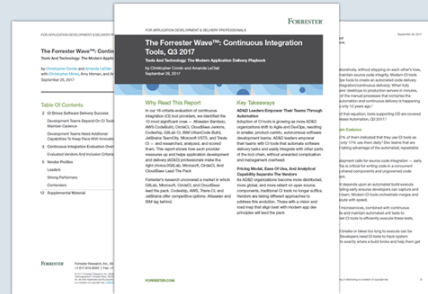
How to get started with Continuous Delivery

If you are wondering how to get started with Continuous Delivery, this article by our CTO Florian Motlik on the Amazon Web Services Blog will be a very helpful read: [Five Tips to Get Started with Continuous Delivery](#).

Forrester Ranks Codeship amongst Top Continuous Integration Vendors

Thousands of customers from over 80 countries are using Codeship every day. Forrester evaluated Codeship and concluded: "Codeship sees the future, and it's Docker."

[Download Report](#)



If you are ready to give Continuous Integration and Delivery a try in your projects feel free to sign up for a free Codeship account today! Simply log in via your GitHub, BitBucket or GitLab account.

[SIGN UP FOR FREE](#)

Continuous Integration Resources

We have compiled a list of resources for you to get started with Continuous Integration and Delivery and also dig deeper if you are more into the topic. You should definitely check out our [Codeship Resources Library](#) where you can find free eBooks, videos, and guides.

Check out our Resources Library

Enjoying our blog? Did you know we are constantly writing new **books** and **courses** for you to **download for free**. Head over to our homepage and have a look around.

[See free books and courses](#)



Books

- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation – by Jez Humble and David Farley:<http://www.amazon.com/gp/product/0321601912>
- Continuous Integration: Improving Software Quality and Reducing Risk – by Paul M. Duvall, Steve Matyas and Andrew Glover:<http://www.amazon.com/gp/product/0321336380>
- The Agile Maturity Model Applied to Building and Releasing Software – by Jez Humble and Rolf Russel:<http://info.thoughtworks.com/agile-maturity-model-applied-building-and-releasing-software.html>
- Recipes for Continuous Database Integration – by Pramod J. Sadalage:<http://www.amazon.com/Recipes-Continuous-Database-Integration-ebook/dp/B000RH0EI4>
- Clean Code – by Robert C. Martin:<http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882>
- The Phoenix Project – by Gene Kim:<http://itrevolution.com/books/phoenix-project-devops-book/>
- Do Faster Releases Improve Software Quality? – Case Study by Mozilla (PDF)<http://swat.polymtl.ca/~foutsekh/docs/Khomh-MSR-2012.pdf>

Blogs & Websites

- <http://www.martinfowler.com/>
 - <http://www.martinfowler.com/articles/continuousIntegration.html>
 - <http://martinfowler.com/articles/is-tdd-dead/>

- <http://agilemanifesto.org/>
- <http://blog.codeship.com>
 - <https://blog.codeship.com/testing-tutorial/>
 - <http://blog.codeship.com/benefits-of-continuous-integration/>
 - <http://blog.codeship.com/seven-steps-to-continuous-deployment/>
- <http://blog.codeclimate.com/>
- <http://chadfowler.com/>
- <https://codeascraft.com/>
- <https://resources.codeship.com/email-courses/continuous-delivery-crash-course>
- <http://patshaughnessy.net/>
- <https://www.twilio.com/engineering/>
- <http://blog.risingstack.com/>
- <https://blog.engineyard.com/>
 - <https://blog.engineyard.com/2012/continuous-integration>
- <https://www.digitalocean.com/community>
- <https://www.airpair.com/posts>
- <https://github.com/kilimchoi/engineering-blogs>
- <https://www.cloudbees.com/blog>

PLATFORM

[Features](#)[Codeship Basic](#)[Parallel Test Pipelines](#)[Codeship Pro](#)[Request a Demo](#)

COMPANY

[About Us](#)[Team](#)[Blog](#)[Customers](#)[Resources](#)

PRICING

[Codeship Pricing](#)[Codeship Basic](#)[Codeship Pro](#)[Start a Conversation](#)[Request a Demo](#)

CASE STUDIES

[Placester](#)[TravelPerk](#)[Product Hunt](#)[Bannerman](#)

COMPARE

[Codeship Pro vs. CircleCI 2.0](#)[Codeship Pro vs. Travis CI](#)[Codeship Pro vs. Gitlab CI](#)[Codeship Basic vs. CircleCI 1.0](#)

HELP

[Documentation](#)[Community](#)[Start with Codeship Pro](#)[Contact Support](#)

RESOURCES

[eBooks](#)[Webinars](#)[Guides](#)[Email Courses](#)


LEGAL

[Terms of Service](#)

[Privacy](#)

[Security](#)

[Imprint](#)

 855.790.8079

 [Twitter](#)

 [Instagram](#)

 [LinkedIn](#)

 [Facebook](#)

 [YouTube](#)

 [Google +](#)