



CircleCI is a continuous integration and continuous delivery platform that can be used to implement DevOps practices.

The company was founded in September 2011 and has raised \$315 million in venture capital funding as of 2021, at a valuation of \$1.7 billion.

CircleCI is one of the world's most popular CI/CD platforms.

Let us take a case scenario.

Imagine working in a car manufacturing assembly line, and you have an option to choose between the two machines provided.

First Machine

It makes good car parts that achieve the job they are meant to do and require very little to no maintenance.

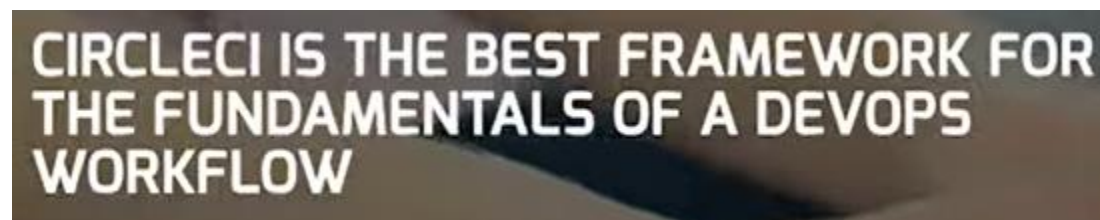
Second machine

It makes perfect car parts all the time and requires much tender love and care.

Which of the two machines would you choose?

This is the exact tradeoff between the Circle CI and Jenkins Server.

Work smarter and not harder.



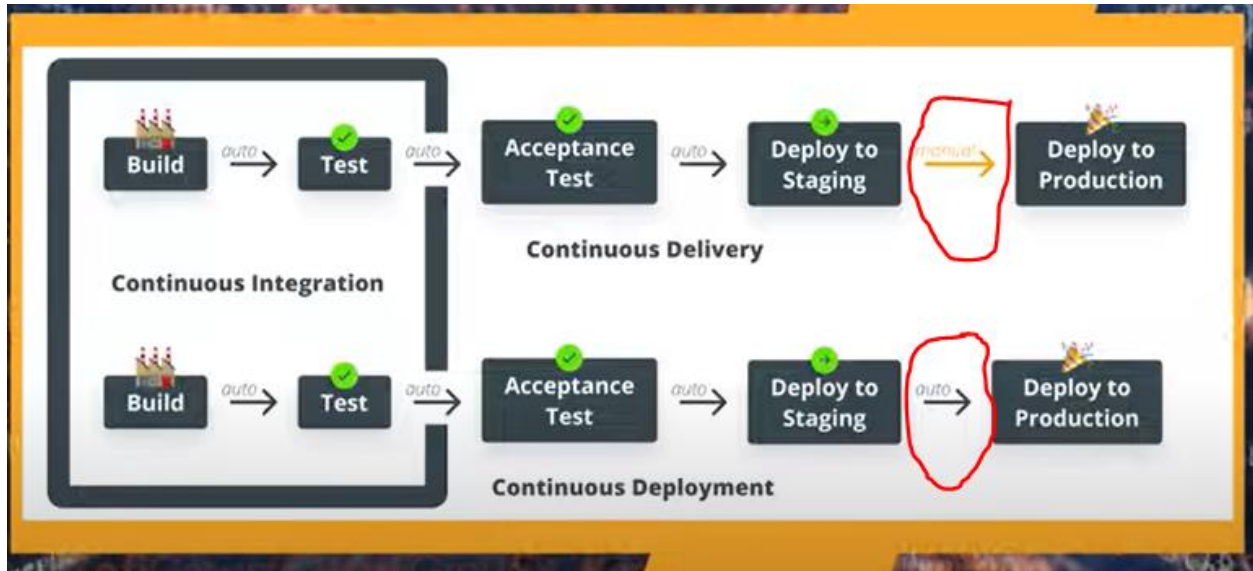
Using Circle CI is to frequently check the small changes and the changes in the lines of codes from the version control system.

Circle CI

HELPS THE DEV TEAM TO INTEGRATE AND VALIDATE CODE CHANGES WITHOUT MUCH DISRUPTION TO FUNCTIONALITY

Continuous integration, continuous delivery, and continuous deployment

Continuous integration



Continuous Delivery

CI ensures that each individual part is made properly, will pass quality assurance tests, and is assembled correctly

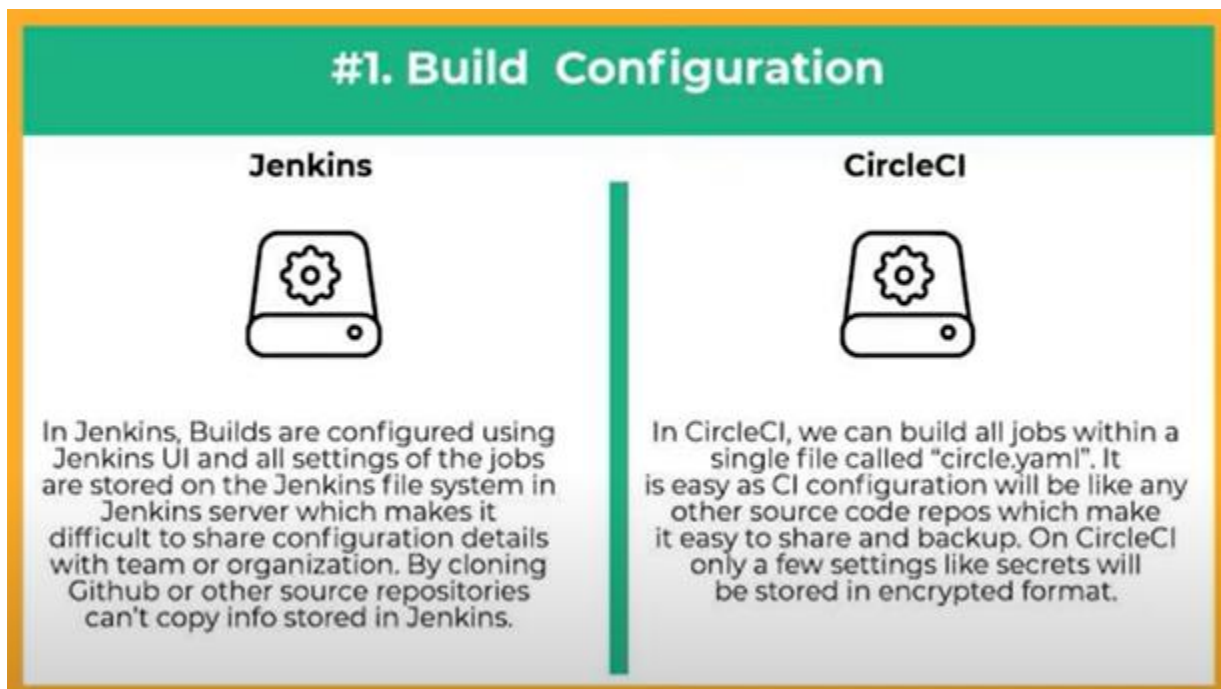
In DevOps, continuous integration is focused on ensuring the new code is written elegantly and tested before it can be merged with the main code in the master/main repository, which is then used to make the final build.

Every DevOps team aims to have both CI and CD processes integrated in the ideal ways possible.

The key is to have your CI and CD implemented so successfully that the handover between the two is a flawless process that takes the click of a button to complete

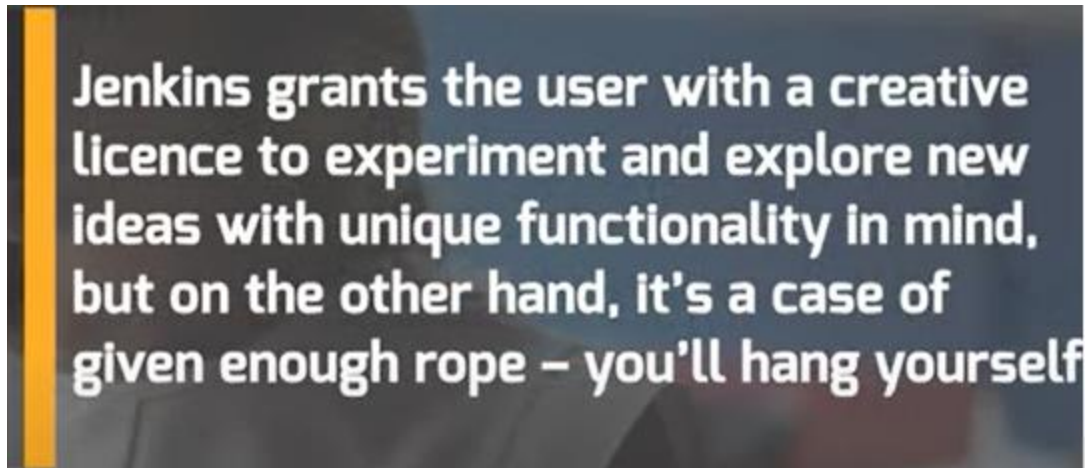
Continuous delivery requires the deployment process to be done manually. In contrast, continuous deployment allows the deployment of the final tested and built code to the live production without any human interventions.

Jenkins vs CircleCI



Jenkins is still a very famous CI/CD tool. It uses third-party plugins to accomplish its tasks. However, as one continues to use more plugins, Jenkins becomes very difficult to maintain and build jobs. Configuration, plugin maintenance, and

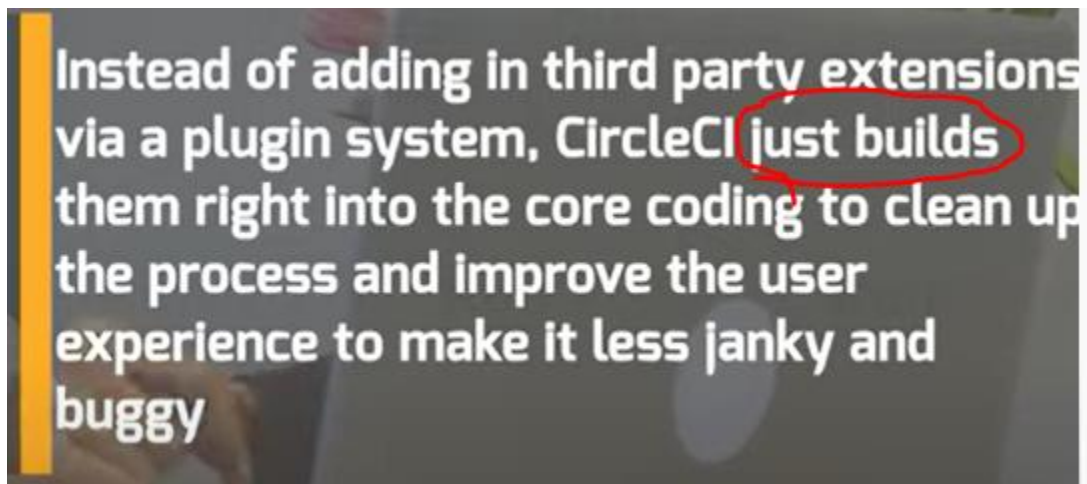
updating processes become more challenging as the software and tasks required to become more complex.



Troubleshooting, analyzing the problems, and finding solutions often takes more time than the software features' actual development, affecting the teams' productivity and ability to move as many features as possible from product backlogs to sprint backlogs.

This means that much time is wasted in troubleshooting and fixing legacy issues related to Jenkins.

CircleCI was created to look at what Jenkins did very well and imitate that and look at what Jenkins did poorly, and aim to improve on this. That simply has the CircleCI was created a decade ago.



Instead of incorporating the third-party plugins, CircleCI just builds the extensions right into the core coding.

Example

```
1  version: 2
2
3  jobs:
4    plan-apply:
5      working_directory: /tmp/project
6      docker:
7        - image: docker.mirror.hashicorp.services/hashicorp/terraform:light
8      steps:
9        - checkout
10       - run:
```

This is a configuration YAML file that is stored in the .circleci and contains the steps and the workflow of the tasks we want to accomplish.

In this project, we use CircleCI and terraform cloud to provision infrastructure on AWS.

If we use Jenkins, we will need to integrate Terraform and AWS cloud bee for credentials plugins.

In CircleCI, we do not need to integrate the extension. We just need to run the extensions as part of the codes.

This makes everything easier and quicker to set up.

Instead of using git-clone, where one needs to integrate git with Jenkins, CircleCI has a unique ability to integrate with Github seamlessly using API tokens.

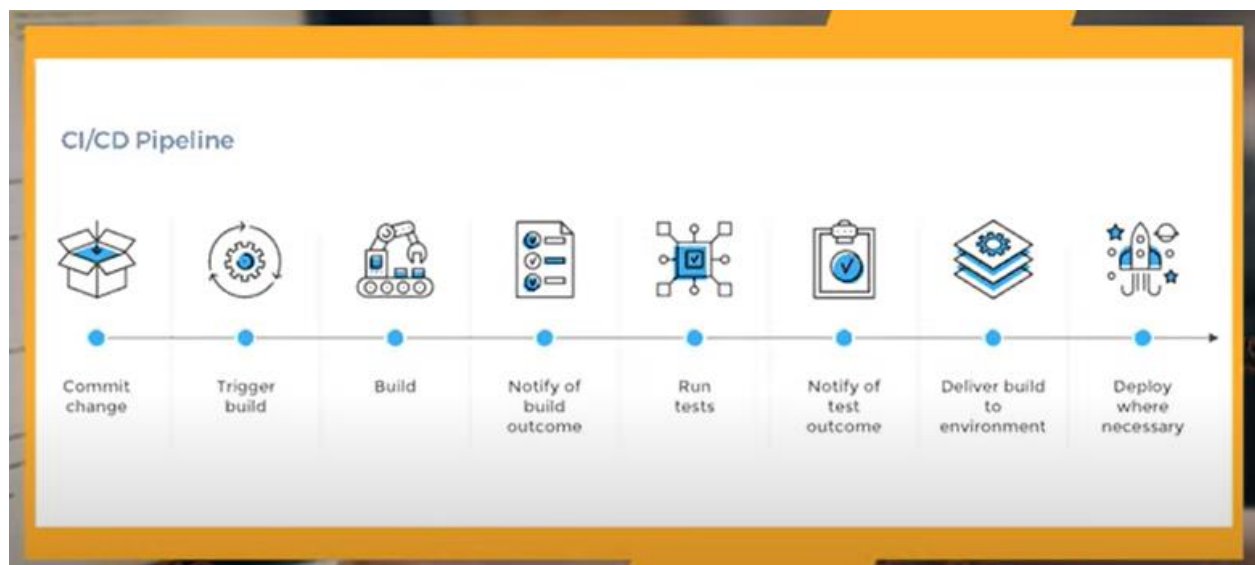
It can literally integrate with any provider or system that has an API, such as



In addition to that, CircleCI has many templates in the form of sharable orbs. This means that somebody somewhere tried to do the same thing you want to do now, and as a result, you have a template you can use instead of starting from scratch.

CircleCI orbs are shareable packages of configuration elements, including jobs, commands, and executors. Orbs do writing and customizing CircleCI config simple. The reusable configuration elements used in orbs are explained fully in the Reusable Configuration Reference.

CircleCI uses



Reducing bugs through testing and building each feature of an application instead of mashing all the features together in the end and struggling with many bugs that could have been caught early on through code writing, building, and testing.

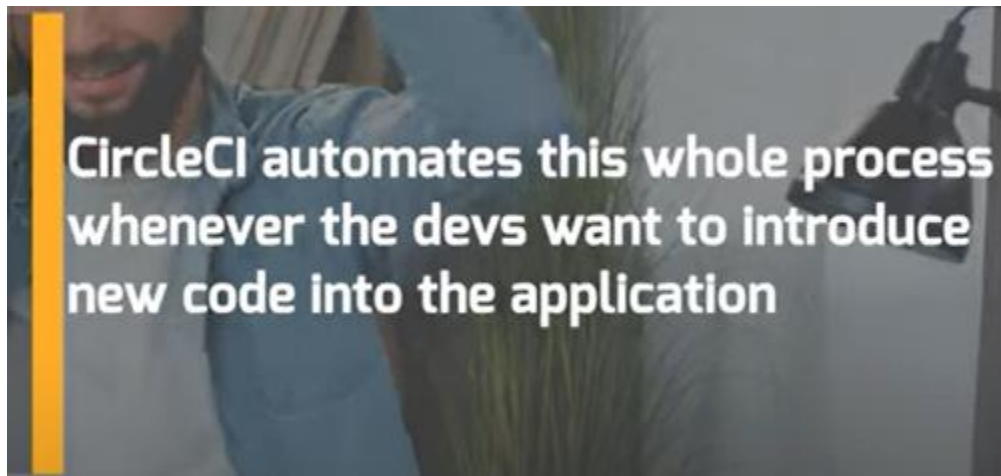
How CI/CD Works



After it passes, it is committed to the master repository, where the build process of the entire code happens to ensure the introduced code does not cause the build to break.

If the build fails or the test fails, the team is forced to evaluate the issues and work on the code again, fixing the issue causing the build to break and tests to fail.

Circles automate this whole process.



This makes it impossible to introduce new code that fails when testing. Unless the code can pass the testing by CircleCI, it will never get introduced to the Master/Main repository.

In case the code passes the CircleCI test, it is introduced to the master/main repository, where it triggers the build process to happen.

If the build fails, the team is forced to look at the outcome and create another branch hoping that the testing and the build steps will pass after fixing the issue.

Every build that passes with flying color is then delivered by CircleCI to the staging area of production, making the CircleCI fulfill the tasks of being a continuous delivery tool.

Not Recommended

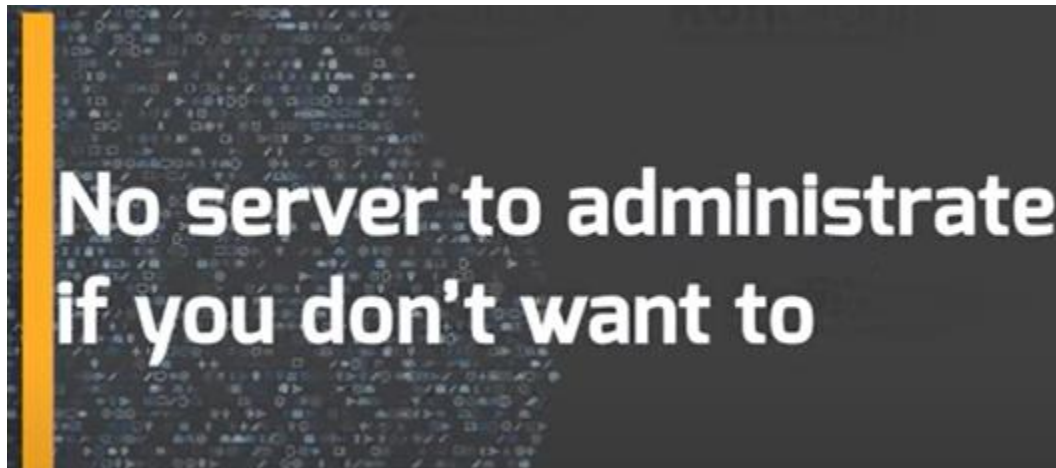
It can also act as a continuous deployment tool by releasing any successful build to the live production environment, such as Kubernetes.

Just like Jenkins, you have the freedom to choose the level of automation that fits your organization.

Cloud-Based System

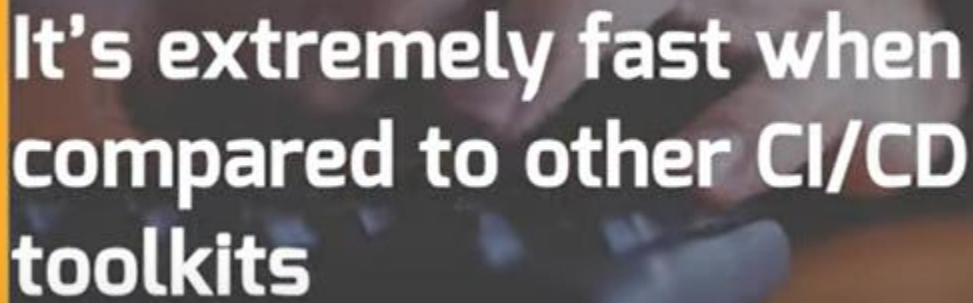
CircleCI is a cloud-based CI/CD tool.

This means no server to administrate if you do want to



Dedicated servers are a thing of the past.

It is beginner-friendly.



**It's extremely fast when
compared to other CI/CD
toolkits**

It is extremely fast because you do not need to set up servers, create an environment or even install packages. You need to create an account, configure the environmental variables, prepare the .circleci folder and create a config.YAML file in it where you will define the steps, containers to install, and the workflow.

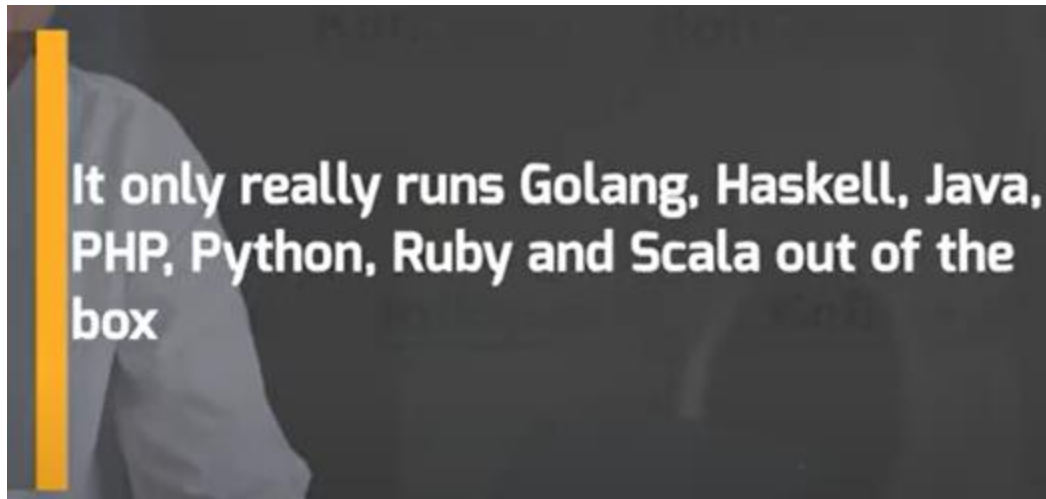
Multitasking and Collaboration



Unlike Jenkins, if you want to carry out the same hundreds of tasks, you will need to build a scalable Jenkins pipeline that can be dynamic or static.

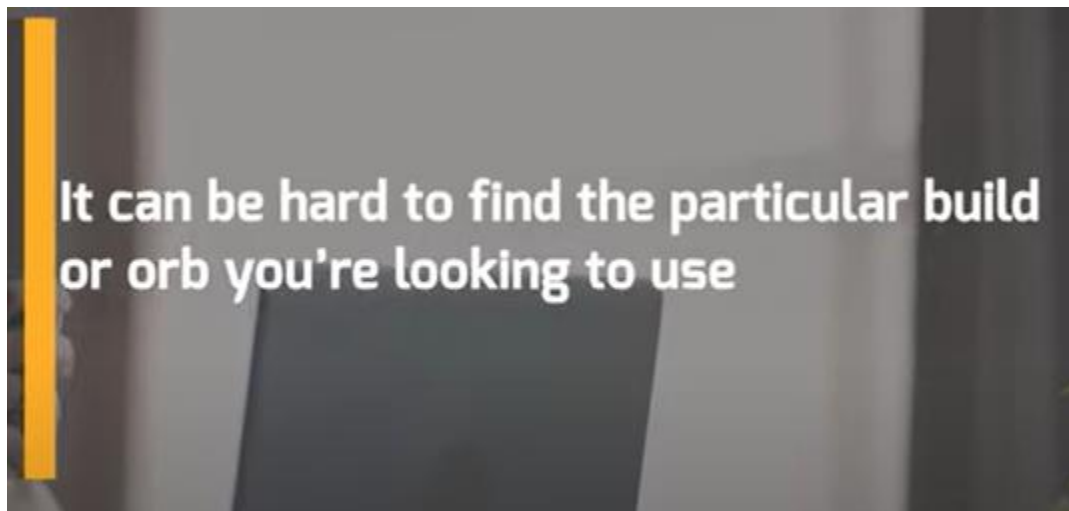
Disadvantages

Although CircleCI can talk to many providers through API tokens, it only allows the following programming languages out of the box. Anything else will require Orbs installation.



It only really runs Golang, Haskell, Java, PHP, Python, Ruby and Scala out of the box

CircleCI has a better UI than Jenkins, but with many Apps enabled, it can be challenging to find a particular build or Orbs



It can be hard to find the particular build or orb you're looking to use

Almost all the major companies have used the CircleCI to build an application. It is a famous CI/CD tool.