CI/CD Pipeline with Jenkins

Introduction to CI/CD

Learning Objectives

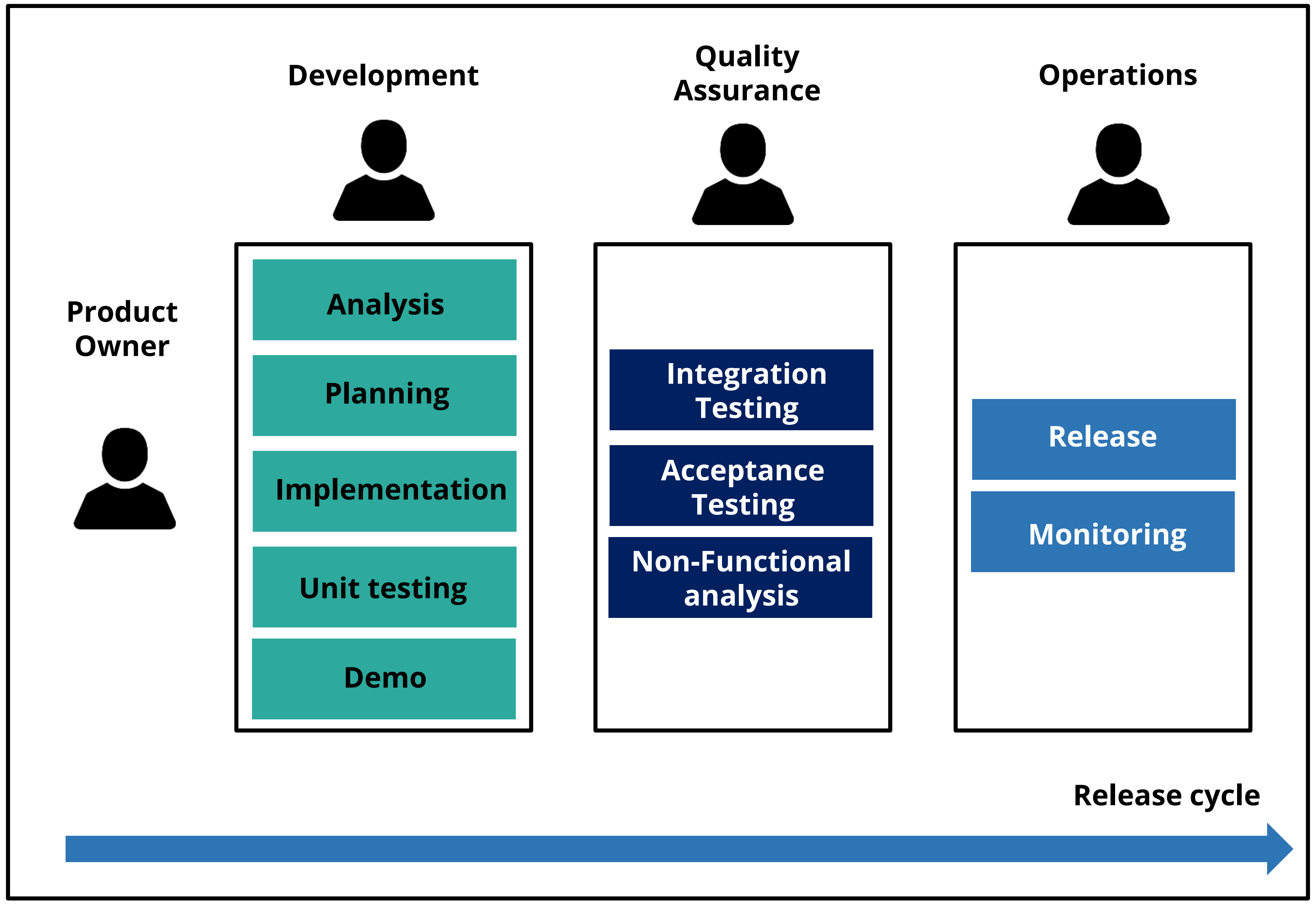
By the end of this lesson, you'll be able to:

- Illustrate the traditional delivery process
- Explain Continuous Integration
- Define Continuous Deployment
- Differentiate between Continuous Deployment and Continuous Delivery
- Describe the automated deployment pipeline

Traditional Software Development

Traditional Delivery Process

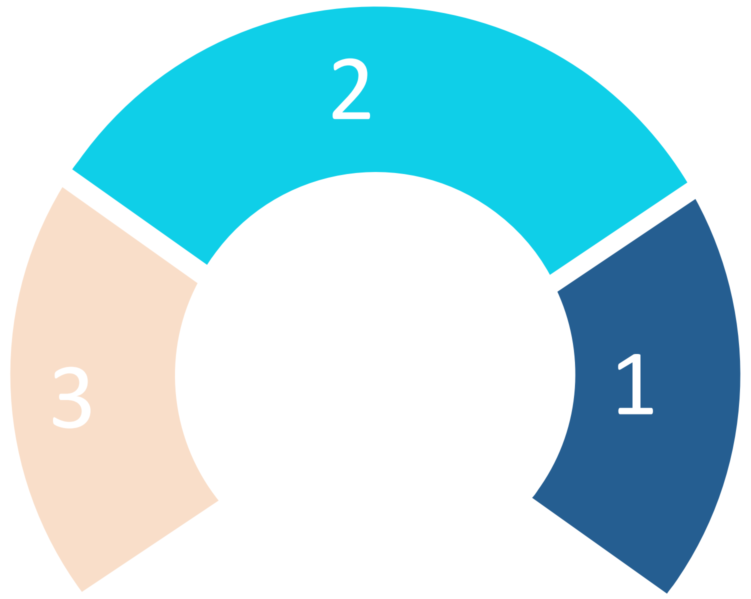
Any delivery process begins with the requirements defined by a customer and ends with the release to production. The diagram below shows the traditional delivery process:



Traditional Delivery Process

The release cycle starts with the requirements provided by the Product Owner. This is followed by three phases, during which the work is passed between different teams.

Quality Assurance
The QA team performs a suite of testing.

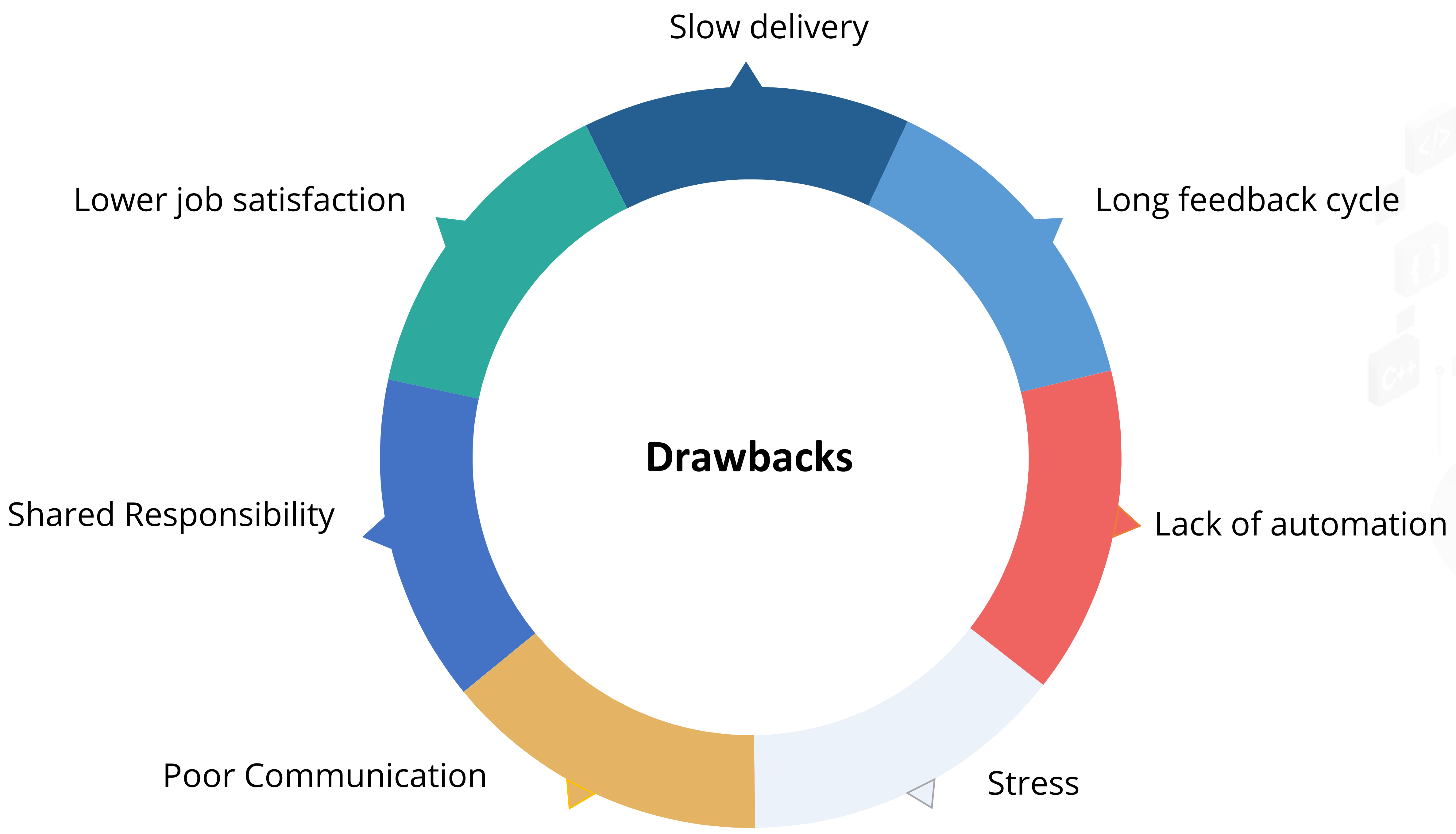


Operations
The Operations team does the release and monitors the production.

Development
The developers work on the product.

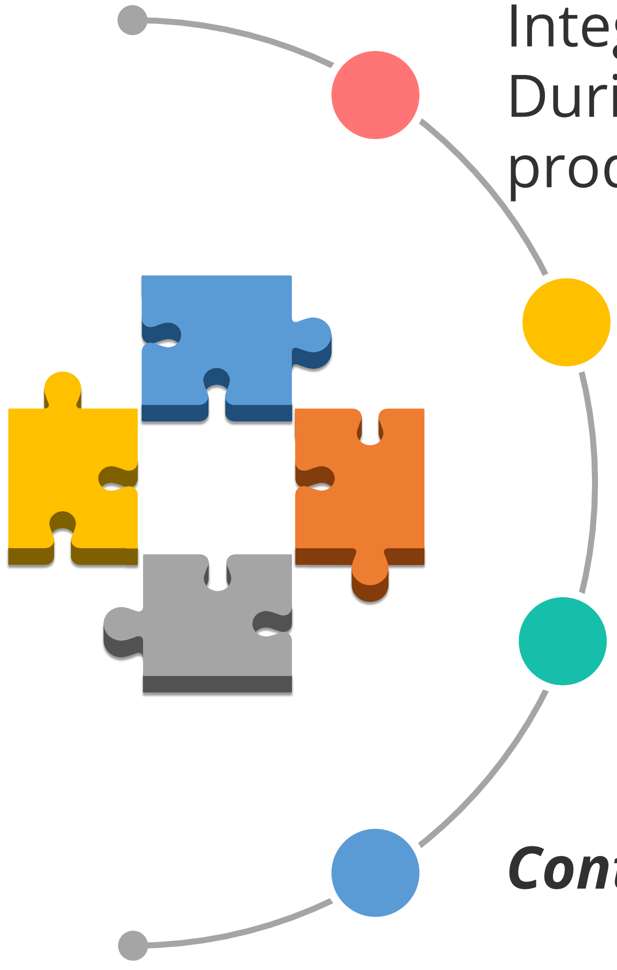
Drawbacks of Traditional Delivery Process

The most significant issues with the traditional delivery process include the following:



Continuous Integration

Introduction



Integration phase in traditional software development cycle was very hectic. During this phase, the code made by individual teams were combined into one product which required extra effort and caused conflicting changes.

It was difficult to anticipate the types of issues that would come up and fix them.

The process involved risks, significant delivery delays, unplanned costs, and unhappy clients.

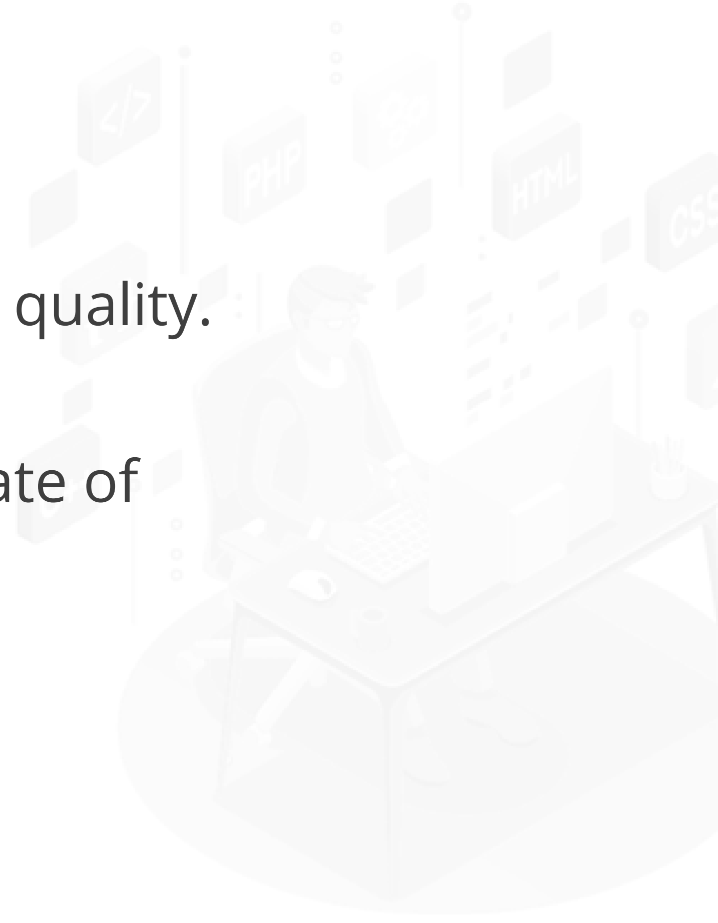
Continuous Integration was introduced to address these issues.

Continuous Integration

Continuous Integration, in its simplest form, involves a tool that monitors your version control system and automatically compiles and tests your application whenever a change is detected.

Advantages of Continuous Integration

- Continuous Integration automatically monitors the health of your codebase, code quality, and code coverage metrics.
- Technical debts are kept down and maintenance costs are low.
- Publicly-visible code quality metrics encourage developers to improve their code quality.
- Automated end-to-end acceptance tests provide a clear picture of the current state of development efforts.



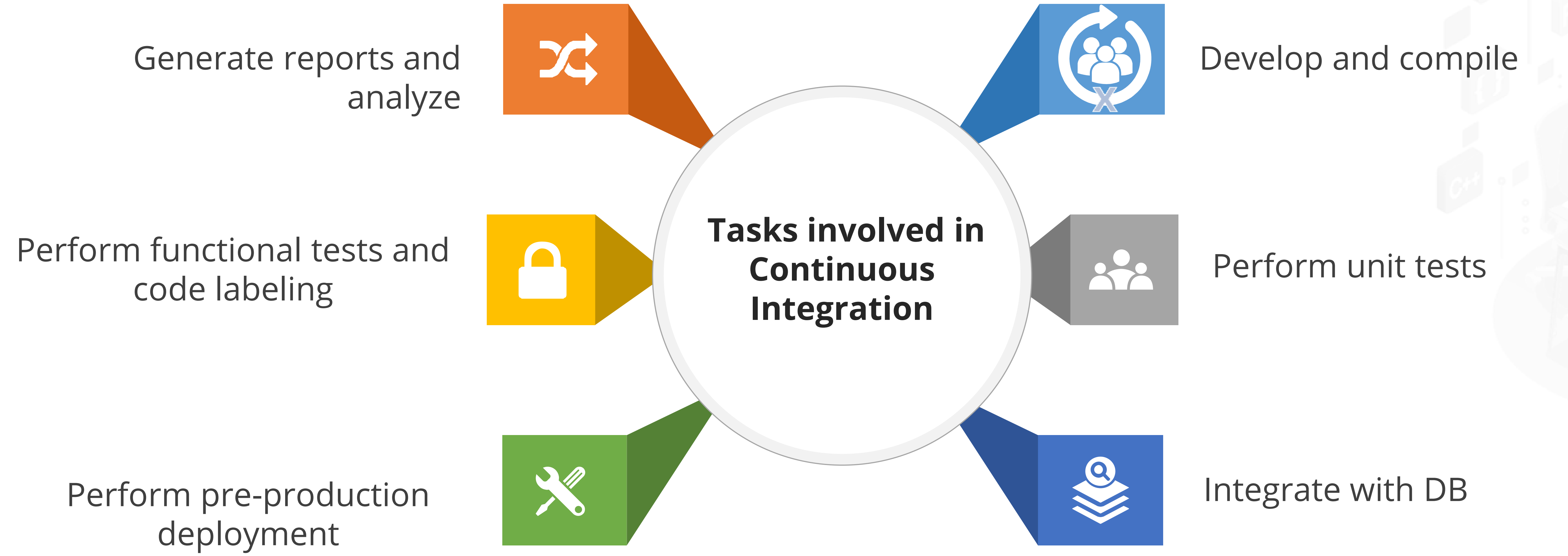
Advantages of Continuous Integration

- Continuous Integration reduces risk by providing faster feedback.
- CI tools are designed to help identify and fix integration and regression issues faster, resulting in fewer bugs and quicker delivery.
- CI helps simplify and accelerate delivery by automating the deployment process.
- Automating the deployment process helps get your software into the hands of the testers and end users faster.

Continuous Integration

Continuous Integration can be defined as a development practice of code integration into a shared repository.

Each integration is verified by an automated build and automated tests. The figure below shows the tasks involved in Continuous Integration.


Tasks involved in Continuous Integration
Develop and compile
Perform unit tests
Integrate with DB
Perform pre-production deployment
Perform functional tests and code labeling
Generate reports and analyze

FULL STACK

Continuous Delivery

Continuous Delivery and Deployment

Continuous Integration lets you deploy the latest version of your application either automatically or as a one-click process.

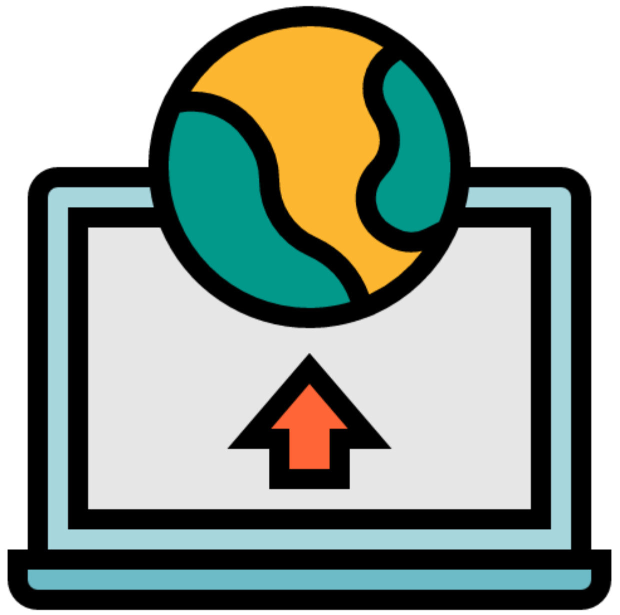
Continuous Delivery is the next step of Continuous Integration. Your code is integrated and tested, and then it is ready to be deployed with one-click.

Automating your deployment eliminates the need for human intervention. Automating the deployment process lets you push every build that passes the tests into production.

The practice of automatically deploying every successful build directly into production is known as Continuous Deployment.

Continuous Delivery

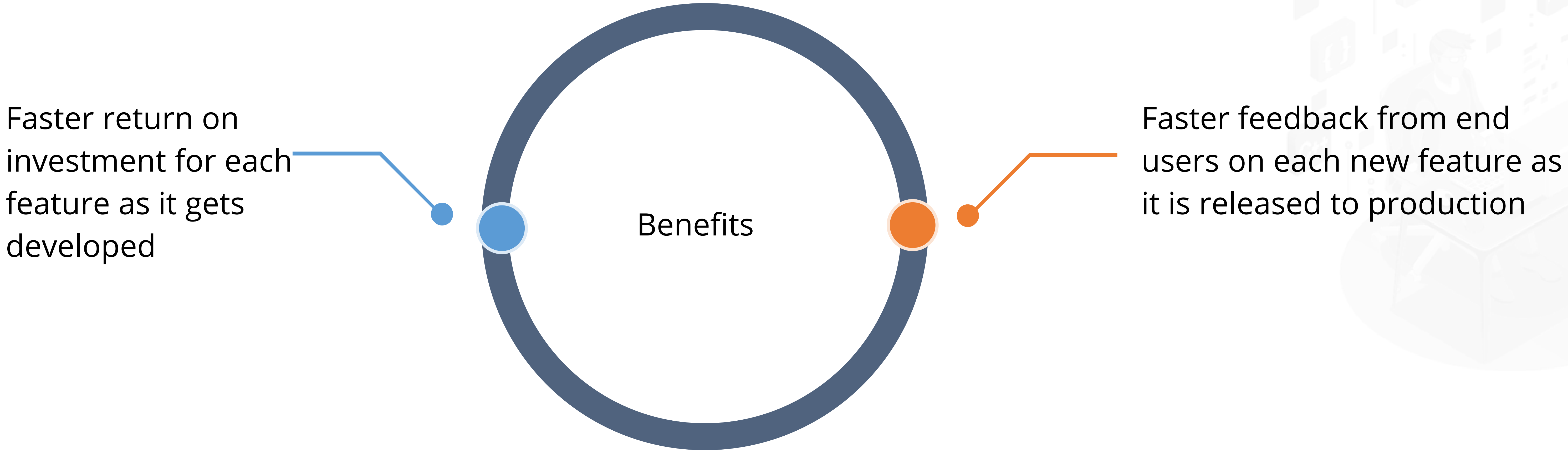
- With Continuous Delivery, any successful build that has passed all the relevant automated tests and quality gates can *potentially* be deployed into production, and be in the hands of the end user within minutes.
- But this process is not **automatic**.
- It is the business, rather than IT that decides the best time to deliver the latest changes.



Continuous Deployment

Continuous Deployment

Continuous Deployment is an extension of continuous integration. It targets to reduce the time between development team writing one new line of code and using it in production.


Faster return on investment for each feature as it gets developed
Benefits
Faster feedback from end users on each new feature as it is released to production

Advantages of Continuous Deployment

Continuous deployment lets us get rid of the tedious release cycle and has the following benefits:

Fast delivery
Customers can use the product as soon as the development is complete.

Fast feedback cycle
Identifying bugs as soon as they are developed, combined with quick rollback strategy, keeps the production stable.

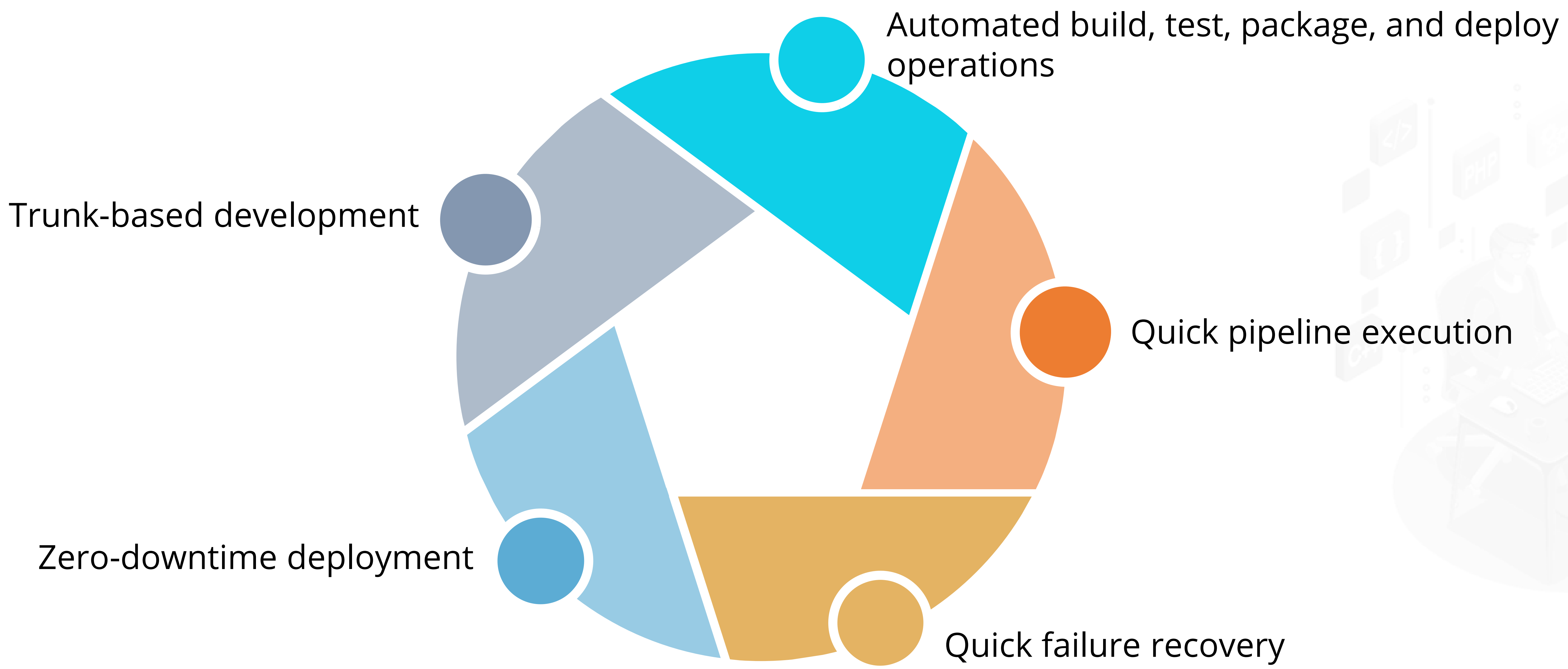
Low-risk releases
If you release on a daily basis, the process becomes repeatable and much safer.

Flexible release options
You can release the software without any additional time or cost spent in case of an immediate release.

Building the Continuous Deployment Process

Prerequisites to CI/CD

Here are a few technical prerequisites for adopting the CI/CD process.



Introducing Tools

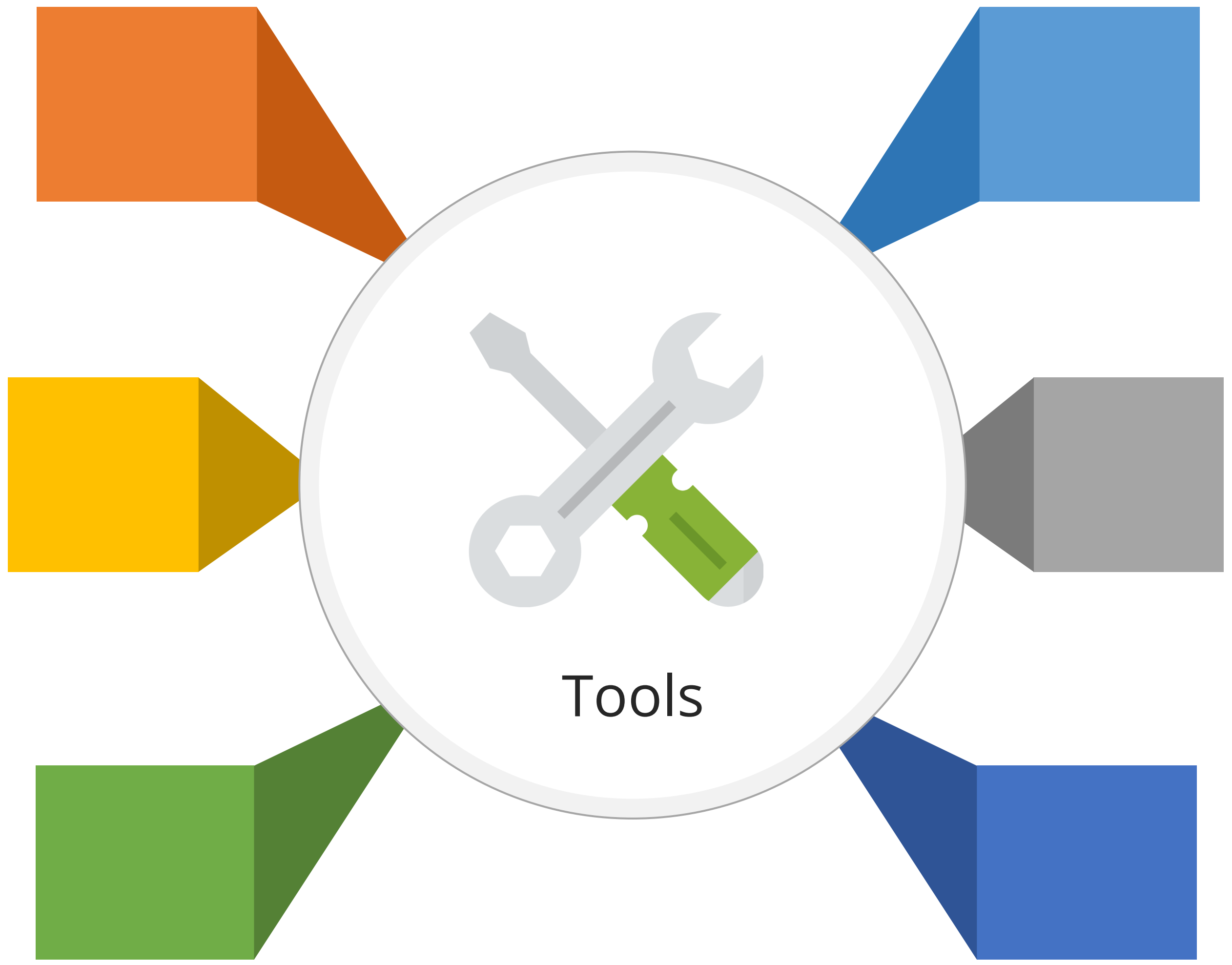
- There are a variety of tools available in the market for performing each of the operations involved in building a Continuous Deployment process.
- Any tool can be replaced with any other tool that plays the same role, depending on your environment.
  - For example: Jenkins can be replaced with Atlassian Bamboo and Chef can be used instead of Ansible.

Continuous Delivery Process Tools

A containerization tool like Docker

A tool for building CI/CD pipelines like Jenkins

A configuration management tool like Ansible



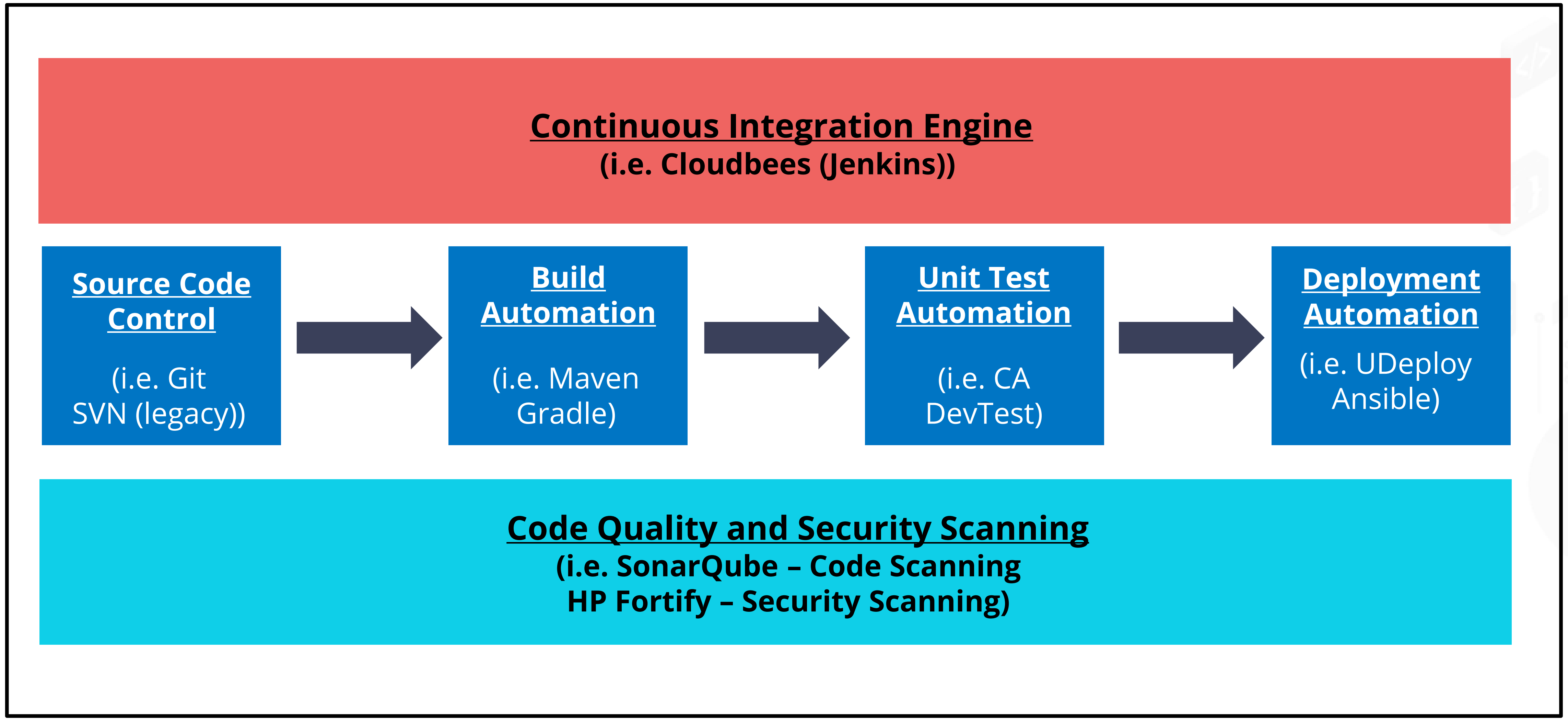
A version control system like Github

A programming language and a corresponding build tool like Java and Maven or Gradle

A testing tool like Cucumber

Continuous Delivery Process Tools

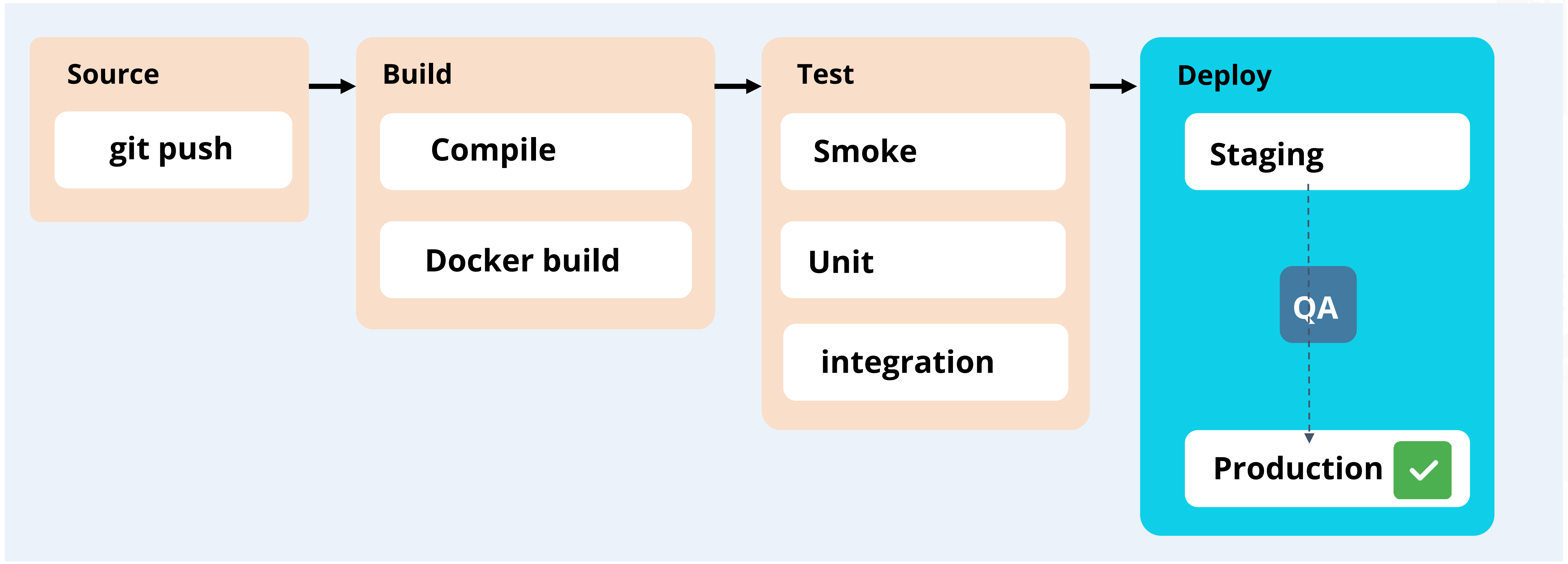
The image below shows a Continuous Delivery pipeline and the tools used along the way:



Automated Deployment Pipeline

Stages of a CI/CD Pipeline

A CI/CD pipeline is essentially a runnable specification of the steps that need to be performed in order to deliver a new version of a software product. A CI/CD pipeline usually has the following stages:



Source Stage

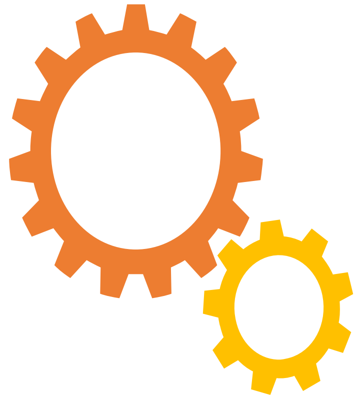
- A pipeline run is usually triggered by a **source code repository**.
- A change in code triggers a notification to the CI/CD tool that runs the corresponding pipeline. Other common triggers include:

Automatically scheduled workflows

User-initiated workflows

Results of other pipelines

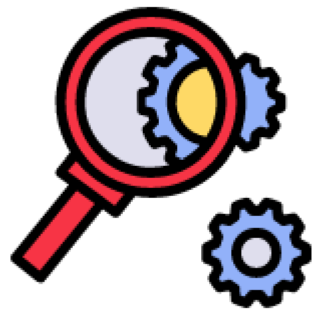
Build Stage



- In the build stage, the source code and its dependencies are combined to build a runnable instance of your product that you can potentially ship to end users.
- Programs written in languages such as Java, C/C++, or Go need to be compiled, whereas Ruby, Python, and JavaScript programs work without this step.
- If the software is deployed using Docker, this stage of the CI/CD pipeline builds the Docker images.
- Failure to pass the build stage is an indicator of a fundamental problem in the configuration of the project.

Test Stage

- In test phase, automated tests run to validate the correctness of the code and the behavior of the product.
- The test stage acts as a safety net that prevents easily reproducible bugs from reaching the end users.
- The responsibility of writing tests falls on the developers, and is best done while writing new code in the process of test- or behavior-driven development.
- Depending on the size and complexity of the project, this phase can last from seconds to hours.



Deploy Stage

Once a runnable instance of the code is built and it passes all predefined tests, it is ready to be deployed.

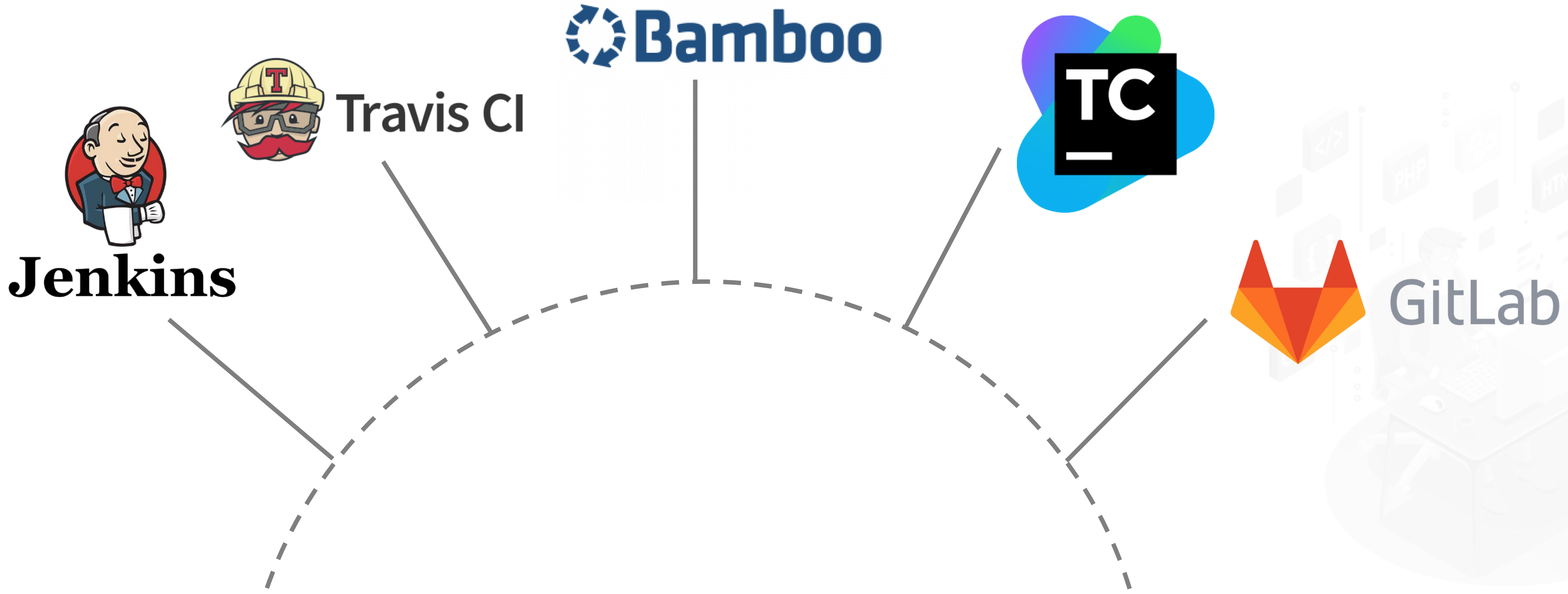
There are usually multiple deploy environments, like a beta or a staging environment to be used internally by the product team and a production environment for end users.

Teams that use the Agile model of development usually deploy work-in-progress manually to a staging environment for additional manual testing and review.

The approved changes are automatically deployed from the master branch to production.

Implementation Of CI/CD

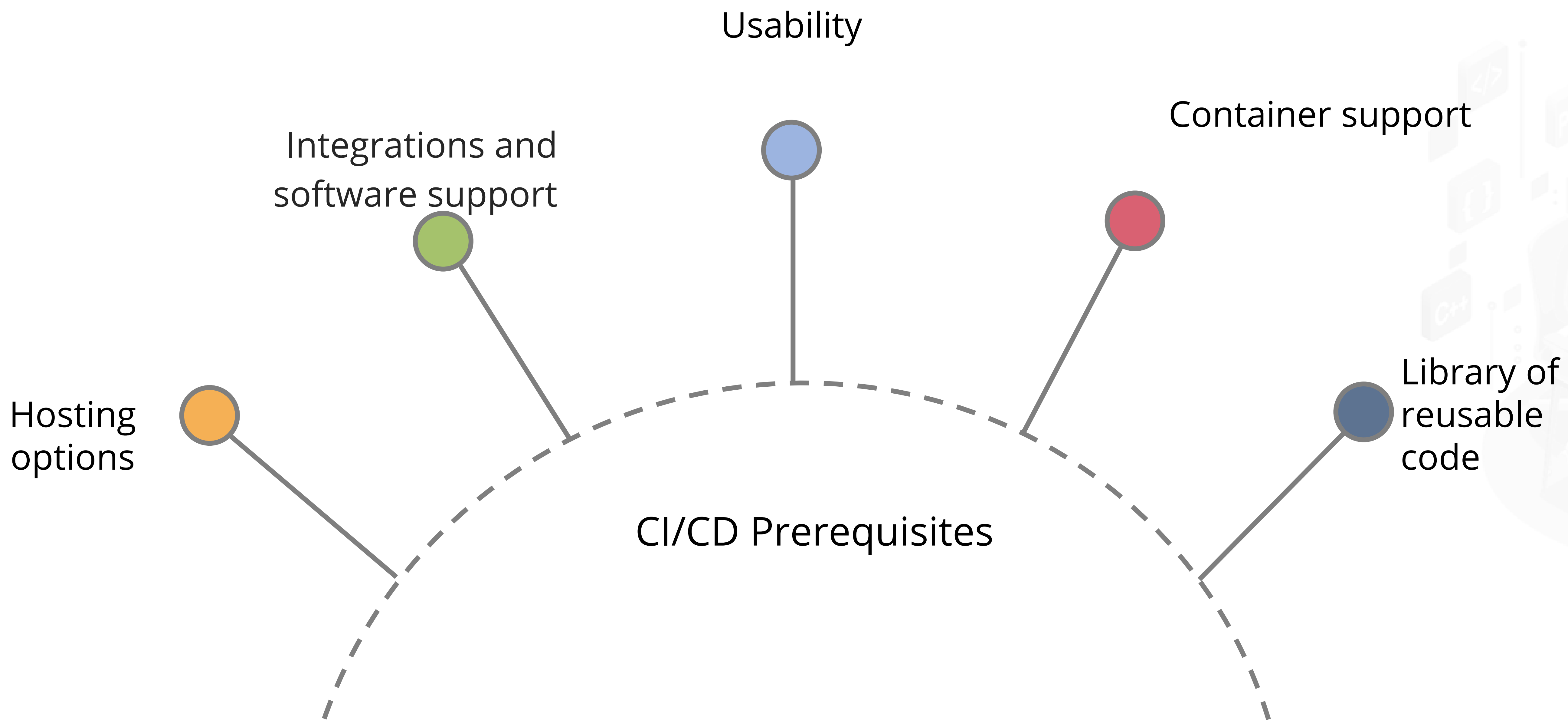
Here is a list of the popular tools available for building CI/CD pipelines:



CI/CD Tool Selection

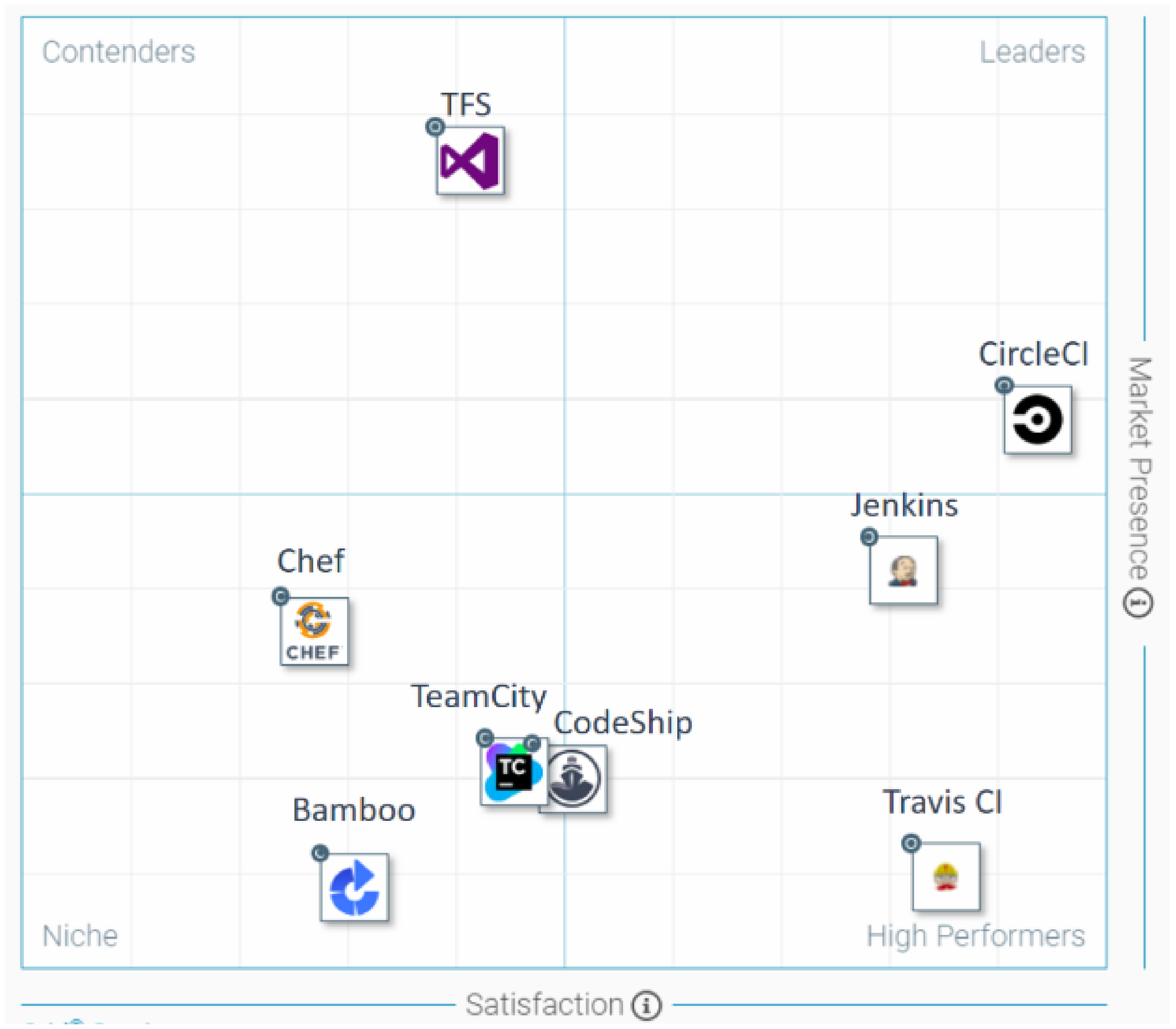
CI/CD Tool Selection

Here are the list of parameters you should consider when selecting a CI/CD tool:



CI/CD Tool Selection

Here is a graph comparing the ratings for various CI/CD tools on StackShare, G2 Crowd, and Slant.co, categorizing them into leader, high-performers, niche, and contenders:



Source: https://www.altexsoft.com/media/2019/02/word-image-55.png

Introduction to Jenkins

- Jenkins is an open-source project written in Java.
- It supports Windows, macOS and other Unix-like operating systems.
- It’s free, community-supported, and is a popular first-choice tool for Continuous Integration.
- Jenkins is primarily deployed on-premises, but it can also run on cloud servers.

Benefits of Jenkins

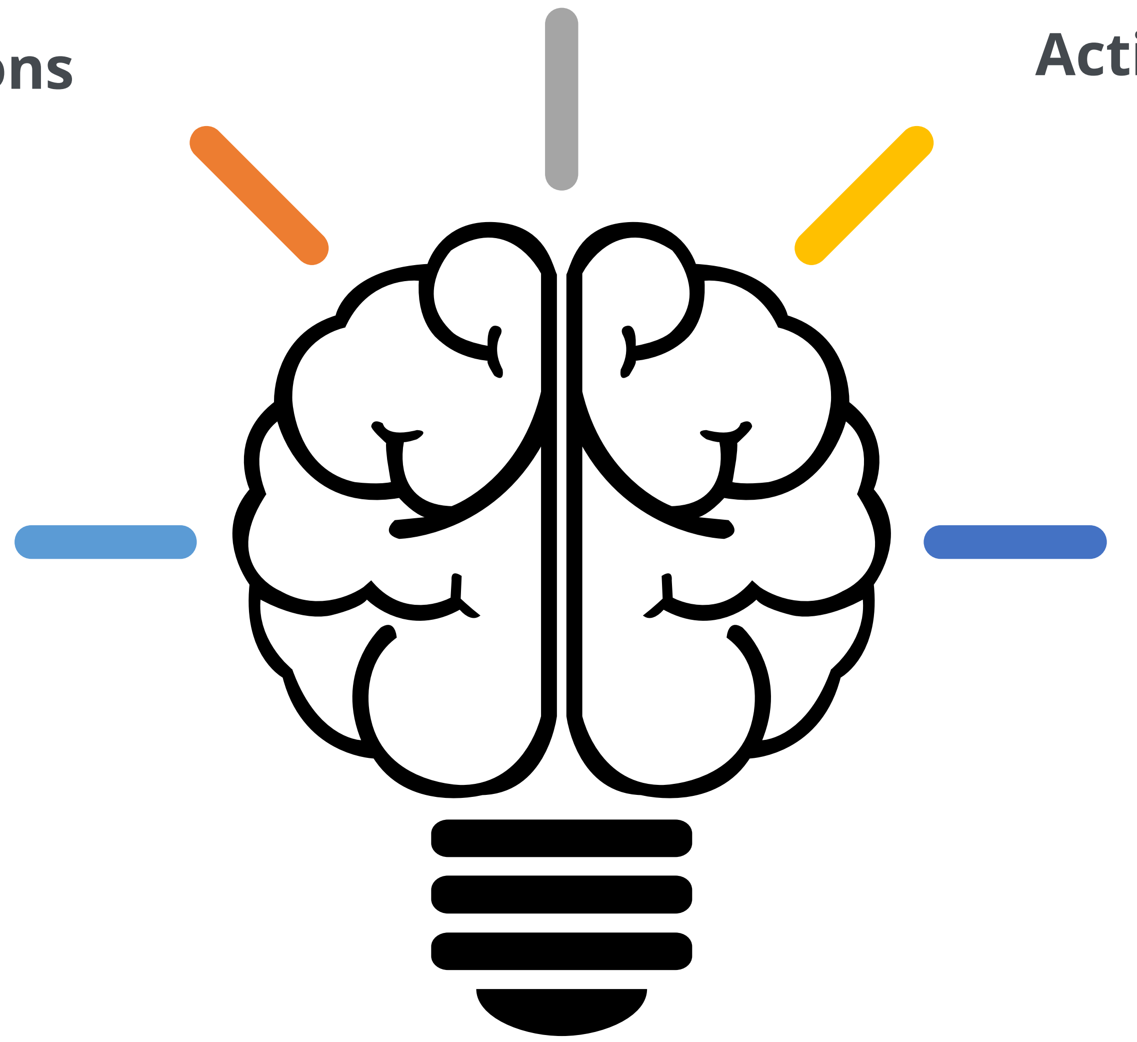
Rich plugins library

Limitless integrations

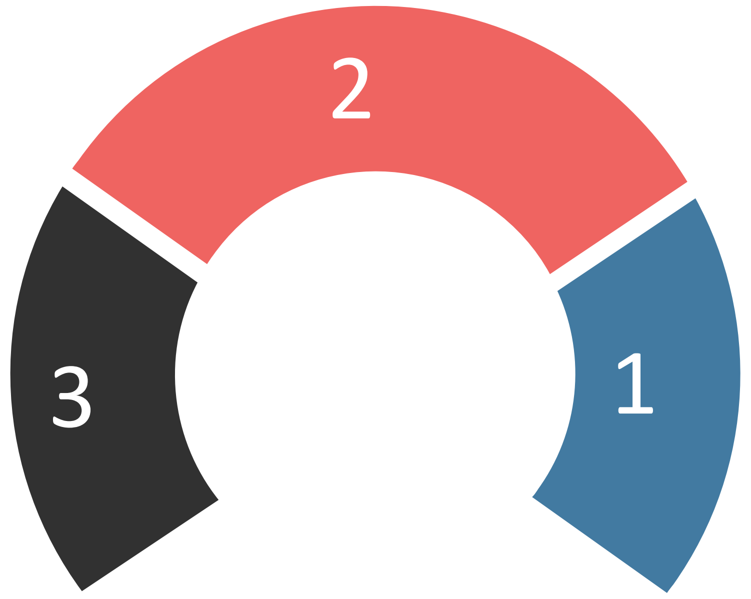
Active community

No expenses required

Distribution of builds and test loads on multiple machines



Drawbacks of Jenkins



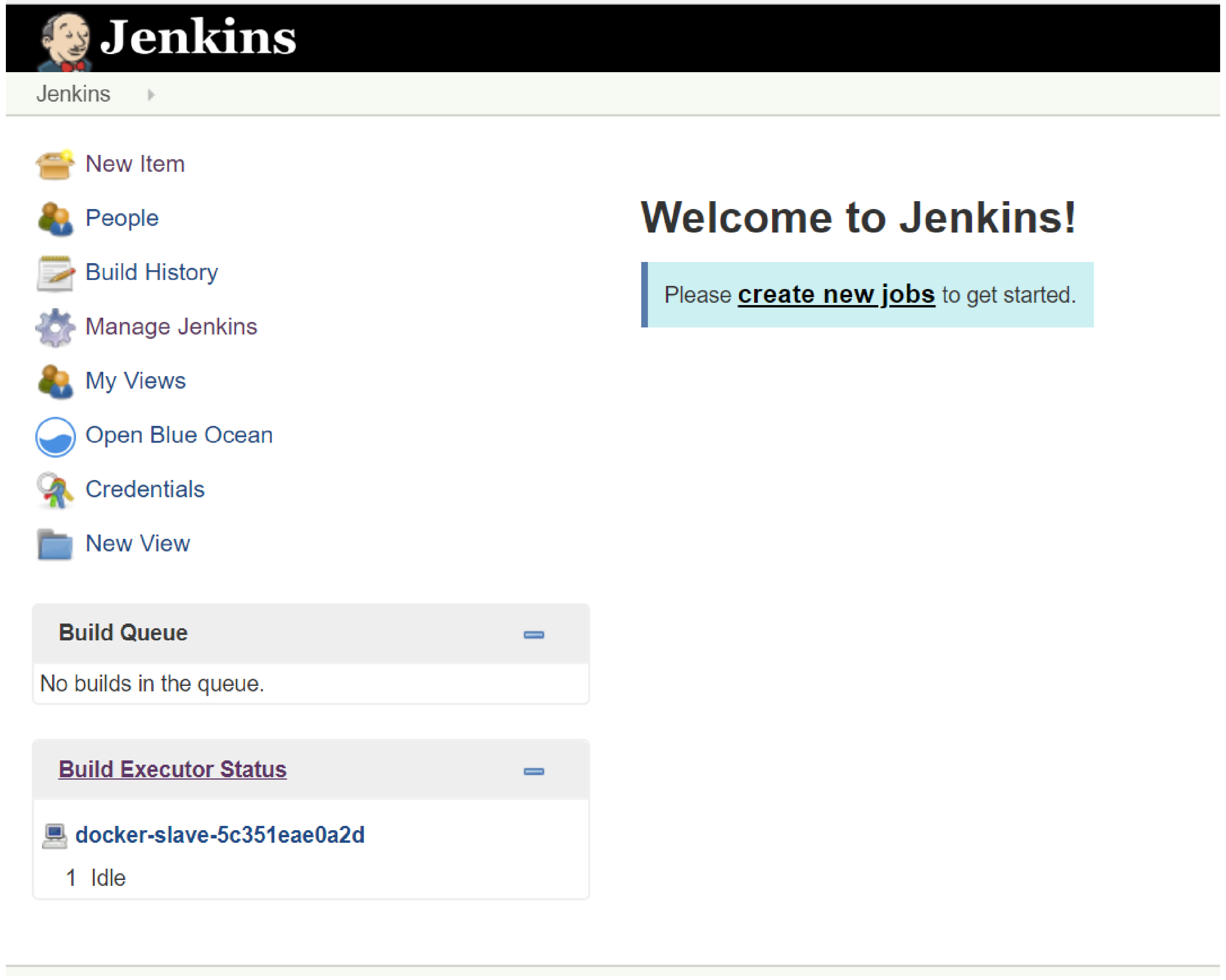
Poor UI
The Jenkins interface seems a bit outdated as it doesn't follow modern design principles.

Manual effort for monitoring
The Jenkins server and its slaves have to be manually monitored to understand interdependencies among the plugins and to upgrade them.

Insufficient Documentation
The documentation sometimes lacks info.

Drawbacks of Jenkins

The picture below shows a screenshot of the Jenkins UI.


Jenkins

Jenkins

New Item
People
Build History
Manage Jenkins
My Views
Open Blue Ocean
Credentials
New View

Build Queue

No builds in the queue.

Build Executor Status

docker-slave-5c351eae0a2d

1 Idle

Welcome to Jenkins!

Please **create new jobs** to get started.

Introduction to TeamCity

TeamCity by JetBrains is a reliable and high-quality CI server.

It has a good number of authentication, deployment, and testing features, and also supports Docker.

It's a cross-platform tool and supports all the recent versions of Windows, Linux, and macOS.

It works with Solaris, FreeBSD, IBM z/OS, and HP-UX.

Benefits of TeamCity

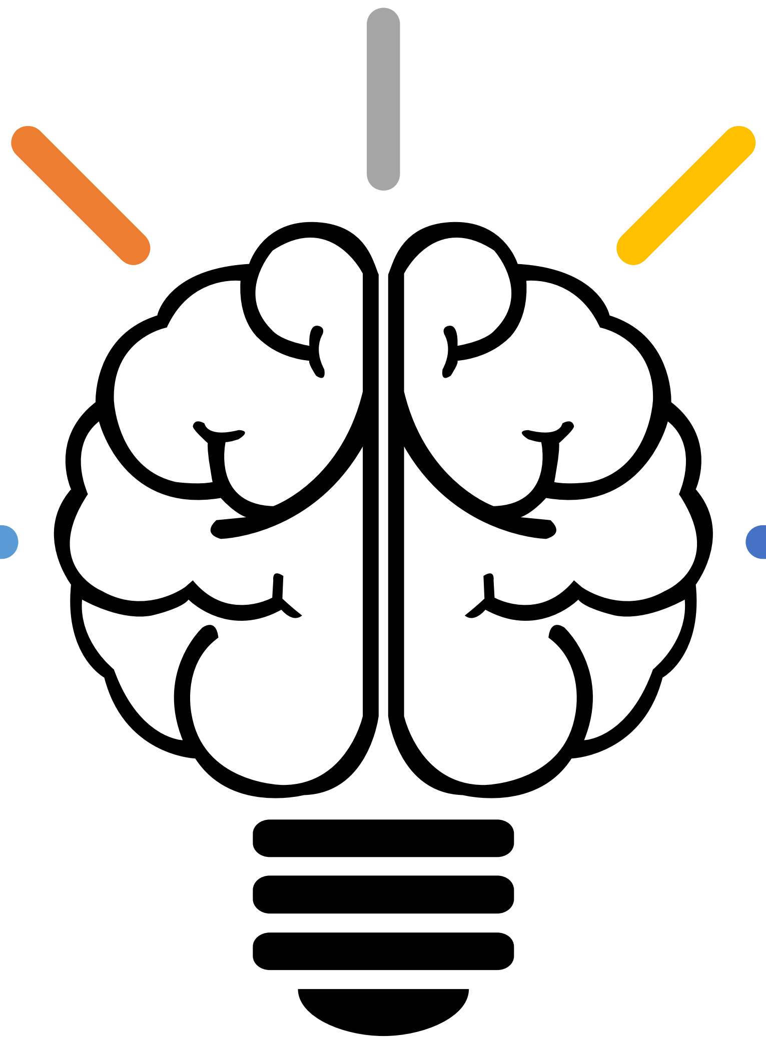
Competent docs

Extensive VSC support

Ease of setup

.NET support

Built-in features



Drawbacks of TeamCity

Steep learning curve

TeamCity is bit complex and overwhelming for newcomer and may take developers some serious study before they are ready to use the tool in production.

Manual upgrading process

Moving from one major version to another is a long process that has to be done manually on your server.

Introduction to Bamboo

Bamboo

Bamboo is a CI/CD tool from Atlassian.

It runs on Windows, Linux, Solaris, and macOS.

Bamboo has features for deployment and release management.

It provides out-of-the-box integration with Atlassian products and integrates natively with other Atlassian products like Bitbucket, Jira, and Confluence.

Benefits of Bamboo

Multiple notification methods
Bamboo Wallboard shows build results on a dedicated monitor and sends build results to your inbox or your Dev chat room via HipChat or Google Talk.

Rich and simple integration
Bamboo supports most major technology stacks, such as CodeDeploy, Ducker, Maven, Git, SVN, Mercurial, Ant, AWS, Amazon S3 Buckets.

Bitbucket Pipelines
Bitbucket Pipelines which are a Git repository management solution from Atlassian can be fully integrated with Bamboo.

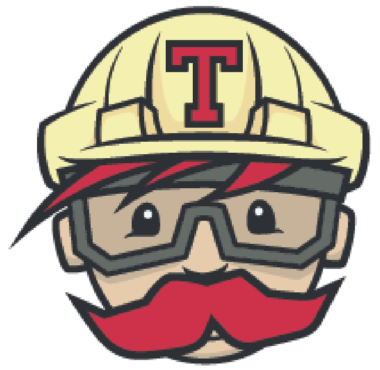
Documentation and support
Bamboo documentation is rich and detailed and Atlassian provides skilled support.

Drawbacks of Bamboo

Poor plugin support
In contrast to Jenkins and TeamCity, Bamboo doesn't support many plugins. There are only 208 apps currently listed on the Atlassian repository.

Complicated first work experience
Some users complain that the setup process of the first deploy task is complex.It takes time to understand all the different options and how to use them.

Introduction to Travis CI



- Travis CI is a mature CI solution with simple GitHub integration.It is one of the oldest CI solutions and has won the trust of many users.
- Travis CI can perform tests on Linux and macOS, but is not recommended for multi-OS testing.
- It has a large, helpful community that welcomes new users and provides a great number of tutorials.
- Travis eliminates the need for a dedicated server, as it is hosted in the cloud. It also offers an on-premises product for companies that want to keep using the same features of the CI tools topped with on-site security needs.

Benefits of Travis CI

Good UI
The user interface is very responsive. Most users say that it's convenient for monitoring builds.

Direct connectivity with GitHub
Travis CI works seamlessly with popular version control systems like GitHub.

Easy setup and configuration
Travis CI requires no installation.
You can begin testing by simply signing up and adding a project.

Backup of the recent build
Whenever you run a new build, Travis CI clones your GitHub repository into a new virtual environment, providing you a backup.

Drawbacks of Travis CI

No CD
Travis CI doesn’t allow for continuous delivery.

GitHub-only hosting
Travis only offers support for GitHub-hosted projects.
The teams that use GitLab or any other alternative are forced to rely on another CI tool.

Introduction to CircleCI

CircleCI is a flexible CI tool that offers up to 16x parallelization.

CircleCI is a cloud-based system but also offers an on-prem solution with security and configuration for running the tool in your private cloud or data center.

It notifies users providing only relevant information via email, HipChat, Campfire, and other channels.

CircleCI provides easy setup and maintenance and a free 2-week macOS trial that allows you to build on both Linux and macOS.

Benefits of CircleCI

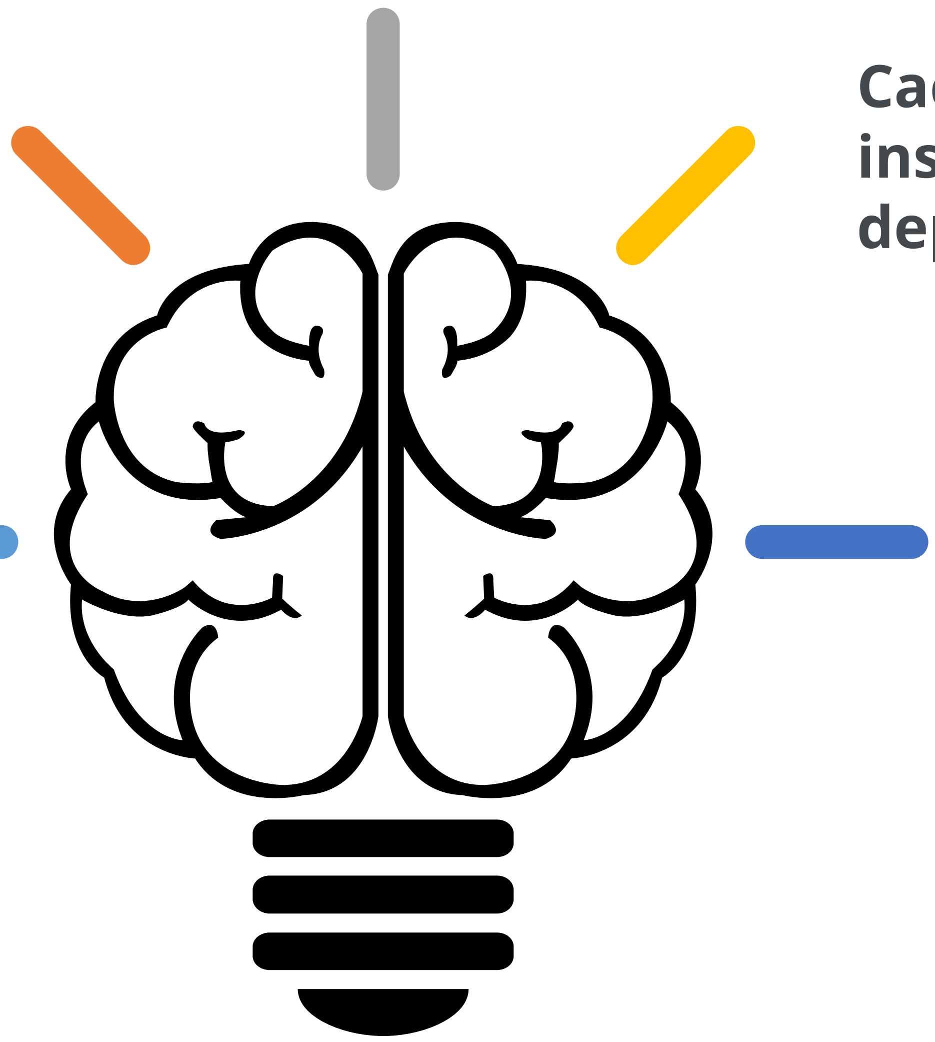
Test Support

High-quality customer support

Caching of requirements installation and third-party dependencies

Simple UI

No need for manual debugging



Drawbacks of CircleCI

Excessive automation	CircleCI changes environment without warning, which may be an issue.
No caching of Docker images	It is not possible to cache Docker images using a private server.
No testing in Windows OS	CircleCI doesn’t yet allow for building and testing in a Windows environment.

Key Takeaways

- Continuous Integration is a development practice of integrating code into a shared repository.
- The practice of automatically deploying every successful build directly into production is known as Continuous Deployment.
- A CI/CD pipeline is essentially a runnable specification of the steps that need to be performed in order to deliver a new version of a software product.
- Popular CI/CD tools include Jenkins, TeamCity, Travis CI, Bamboo, and CircleCI.