SDLC

**What Is SDLC? Understand the Software Development Life Cycle**

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, the SDLC methodology focuses on the following phases of software development:

* Requirement analysis
* Planning
* Software design such as architectural design
* Software development
* Testing
* Deployment

This article will explain how SDLC works, dive deeper in each of the phases, and provide you with examples to get a better understanding of each phase.

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, the SDLC methodology focuses on the following phases of software development:

* Requirement analysis
* Planning
* Software design such as architectural design
* Software development
* Testing
* Deployment

This article will explain how SDLC works, dive deeper in each of the phases, and provide you with examples to get a better understanding of each phase.

## Examples

The most common SDLC examples or SDLC models are listed below.

### Waterfall Model

This SDLC model is the oldest and most straightforward. With this methodology, we finish one phase and then start the next. Each phase has its own mini-plan and each phase “waterfalls” into the next. The biggest drawback of this model is that small details left incomplete can hold up the entire process.

### Agile Model

The Agile SDLC model separates the product into cycles and delivers a working product very quickly. This methodology produces a succession of releases. Testing of each release feeds back info that’s incorporated into the next version. [According to Robert Half](https://www.roberthalf.com/technology/blog/6-basic-sdlc-methodologies-the-pros-and-cons), the drawback of this model is that the heavy emphasis on customer interaction can lead the project in the wrong direction in some cases.

### Iterative Model

This SDLC model emphasizes repetition. Developers create a version very quickly and for relatively little cost, then test and improve it through rapid and successive versions. One big disadvantage here is that it can eat up resources fast if left unchecked.

### V-Shaped Model

An extension of the waterfall model, this SDLC methodology tests at each stage of development. As with waterfall, this process can run into roadblocks.

### Big Bang Model

This high-risk SDLC model throws most of its resources at development and works best for small projects. It lacks the thorough requirements definition stage of the other methods.

### Spiral Model

The most flexible of the SDLC models, the spiral model is similar to the iterative model in its emphasis on repetition. The spiral model goes through the planning, design, build and test [phases](https://stackify.com/sdlc-phases-identify-problems/) over and over, with gradual improvements at each pass.

## Benefits of the SDLC

SDLC done right can allow the highest level of management control and documentation. Developers understand what they should build and why. All parties agree on the goal upfront and see a clear plan for arriving at that goal. Everyone understands the costs and resources required.

Several pitfalls can turn an SDLC implementation into more of a roadblock to development than a tool that helps us. Failure to take into account the needs of customers and all users and stakeholders can result in a poor understanding of the system requirements at the outset. The benefits of SDLC only exist if the plan is followed faithfully.

Want to improve application quality and monitor application performance at every stage of the SDLC? [Try out Stackify’s Retrace tool](https://stackify.com/retrace/) for free and experience how it can help your organization at producing higher-quality software.

# 4 Values and 12 Principles of the Agile Manifesto

Agile Manifesto is the foundation of most modern methodologies of project management. It has four core values supplemented by 12 principles. [Project managers](https://kissflow.com/project/how-project-managers-stay-organized-at-work/) make use of these principles to deliver extraordinary products, with both value and quality, while staying within the given [constraints of the project](https://kissflow.com/project/project-management-constraints/).

The actual document consisting of the values and principles of the Agile approach was developed in [2001](https://www.agilealliance.org/) by a group of 17 software engineers at a ski resort in Utah. The ‘Agile Alliance’ was created to challenge the status quo and completely change the approach to problem-solving and [project management](https://kissflow.com/project/guide-to-project-management/).

What started as a guide for software development has now become a globally accepted philosophy for [handling projects](https://kissflow.com/project/what-is-a-project/). Today, Agile processes is not only used by software development teams but by almost every function of a business. Many of the original members of the Agile Alliance went on to develop other popular frameworks like Scrum methodology, Kanban methodology, Crystal, and Integrated Agile based on the Agile methodology.

Unlike other project management methodologies, [Agile methodology](https://kissflow.com/project/agile/agile-project-management-methodology/) gives complete freedom to the project team. There is no set of rules, frameworks, procedures, or hierarchy that must be followed. As long as you are adhering to the Agile values and Agile principles, you are good to go.

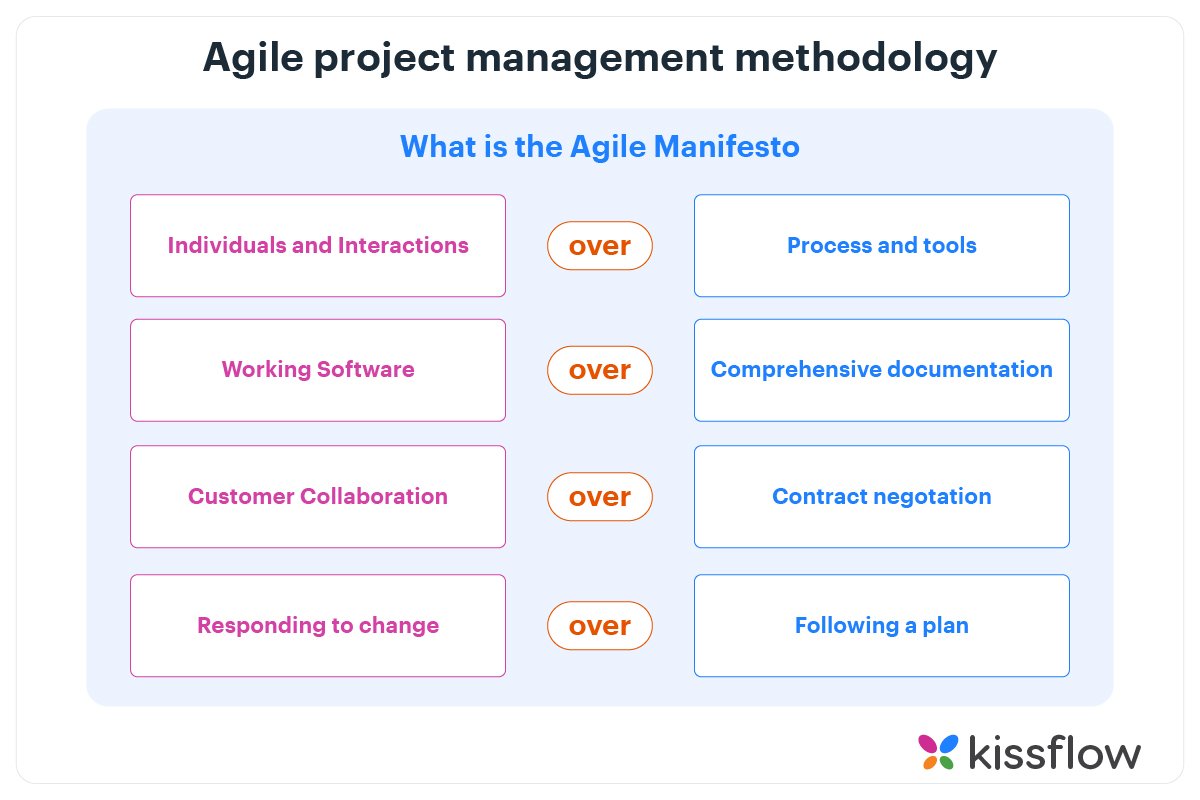
## Why was the Agile methodology developed?

In the late ’90s, both workers and clients were frustrated with the existing [project management methodologies](https://kissflow.com/project/project-management-methodologies-and-frameworks/) like the Waterfall. The disparity between delivered products and the client’s requirements were huge. [Delays in the project](https://kissflow.com/project/how-to-avoid-project-delays/) were common and many projects were either canceled or left the clients unsatisfied with the performance.

Teams were more concerned about proper documentation rather than the customer’s requirements. Take the software industry for example. **Traditional methods were unable to keep up with the ever-changing demands of the customers.** Developers were unable to fully use the versatility of developing software.

There was a consensus that companies are focusing on the wrong things and wasting their efforts. There was a need for an innovative, modern methodology that can quickly adapt to the changing circumstances and give certain freedom to the project teams in order to increase their efficiency and performance.

## 4 values of Agile Manifesto



**That is, while there is value in the items on**  
**the right, we value the items on the left more.**

#### 1. Individuals and interactions over processes and tools

This value of the Agile manifesto **focuses on giving importance to communication with the clients**. There are several things a client may want to ask and it is the responsibility of the team members to ensure that all questions and suggestions of the clients are promptly dealt with.

#### 2. Working product over comprehensive documentation

In the past, more focus used to be on proper documentation of every aspect of the project. There were several times when this was done at the expense of the final product. The Agile values dictate that **the first and foremost duty of the project team is completing the final deliverables** as identified by the customers.

#### 3. Customer collaboration over contract negotiation

Agile principles **require customers to be involved in all phases of the project**. The [Waterfall approach](https://kissflow.com/project/agile/traditional-vs-agile-project-management/) or Traditional methodologies only allow customers to negotiate before and after the project. This used to result in wastage of both time and resources. If the customers are kept in the loop during the development process, team members can ensure that the final product meets all the requirements of the client.

#### 4. Responding to change over following a plan

Contrary to the management methodologies of the past, Agile values are against using elaborate plans before the start of the project and continue sticking to them no matter what. Circumstances change and sometimes customers demand extra features in the final product that may change the [project scope](https://kissflow.com/project/project-scope-management/). In these cases, project managers and their **teams must adapt quickly in order to deliver a quality product and ensure 100% customer satisfaction**.

## 12 Principles of Agile Manifesto

#### 1. Customer satisfaction through continuous delivery of the product

In the case of [traditional management methodologies](https://kissflow.com/project/agile/traditional-vs-agile-project-management/), customers get to see the product only after completion and when several tests and quality checks have been performed. This not only keeps the customers in dark but also makes it problematic for the team members to introduce any changes in the product.

In order to keep the customers happy, it’s important to continuously engage them with a working version of the product. Show small increments every [sprint planning](https://kissflow.com/project/agile/scrum-sprint-planning/) and make changes as required.

#### 2. Divide large chunks of work into smaller and achievable tasks for quicker completion and easier integration of changes

Handling a huge and complex task would be both time and energy-consuming while [managing project tasks](https://kissflow.com/project/team/guide-to-task-management/). A better way is to divide the task into smaller parts that can be easily completed. The customers would always be kept in the loop and it would be easier for the team members to identify potential bottlenecks and handle any potential delays.

#### 3. Adhere to the decided timeframe for the delivery of a working product

The Agile philosophy favors a smaller time frame and delivers working software frequently. This iterative process requires team members to continuously improve their performance.

#### 4. All stakeholders must frequently collaborate to ensure that the project is going in the correct direction

A major problem associated with traditional project management methodologies is that the [project stakeholders](https://kissflow.com/project/project-stakeholder-management/) are often oblivious to the development stages of the project. The Agile principles encourage all stakeholders to remain involved in all stages of the project in order to ensure constant feedback and a valuable end product.

#### 5. Create a supportive environment to motivate team members and encouraging them to get the job done

It is the responsibility of the project manager to create a motivating environment and support where members are not afraid to voice their opinions and give suggestions for the betterment of the team’s performance. Giving the team members the confidence and motivating them would cause their overall performance to improve significantly which will ultimately be better for the project.

#### 6. Prefer face-to-face communication over other methods

In the Agile manifesto, a lot of importance is given [to effective communication](https://kissflow.com/project/communication-in-project-management/) between the involved parties. For effective communication, methods like memos and email are not preferred and more importance is given to face-to-face communication. This is now easier because of the advances in communication technologies. All stakeholders can now have a meeting through video conferencing rather than coming to the office for a short meeting.

#### 7. Working software is the primary measure of progress

The only factor to measure success is the delivery of a working product that satisfies the customer. Before Agile, there were many measures of success and that resulted in a drop in the quality of [the final product](https://kissflow.com/project/project-deliverables/).

#### 8. Try to maintain a constant pace of development

A repeatable and iterative pattern should be established where sustainable development of the project takes place at a constant rate. This prevents the team members from being overwhelmed and over-stressed, while the project keeps on progressing at an acceptable rate.

#### 9. Maintain the quality of the product by paying attention to technical details

Providing value to the customer is the primary objective of any [Agile team](https://kissflow.com/project/agile/agile-team/). It’s extremely important to have a multi-skilled team that can handle all the technical aspects of the project and provides the opportunity for continuous improvement.

#### 10. Maintain simplicity

In each time box, the tasks at hand should be the main focus of all team members. Too much planning and adding extra features to the product should be avoided during the development.

#### 11. Promote self-organization in the team

A [self-organized team](https://kissflow.com/project/agile/importance-of-self-organizing-teams/) with decision-making powers would simply perform better because the responsibility of satisfying the customers will on the team members, rather than a single project manager.

#### 12. Regularly reflect on your performance for continuous improvement

Agile methodologies stand on the concept of iteration, where teams learn from their past mistakes and continuously improve their performance. Project managers should promote sessions where the whole team reflects on their performance and discuss ways to improve their technical and management skills.

## Getting started with Agile

Although Agile methodologies are not too complicated, you still needed some help from experts in the past. But now, you can shift to Agile principles and values by using different software available in the market.

[Kissflow Project](https://kissflow.com/project/) is one of those applications which is intuitive for those who do not know much about the Agile Methodology. It is a free project management software and offers a simple interface and [Kanban boards](https://kissflow.com/project/agile/kanban-board-software/) to organize all your tasks. You can add team members, collaborate with them on the board, and track progress easily.

## What are the three scrum roles?

A Scrum team consists of three distinct roles: the Scrum Master, the product owner, and development team members. While there is only one Scrum Master and one product owner, there are generally several development team members.

Scrum teams are small; the Scrum Guide recommends ten or less total members to make sure communication and productivity is optimal [[3](https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100)]. Let’s take a closer look at what the responsibilities are for each of these roles.

**Read more:** [What Is a Scrum Master (and How Do I Become One)?](https://www.coursera.org/articles/what-is-a-scrum-master)

## 1. Scrum Master

A Scrum Master is the person responsible for making sure a Scrum team is operating as effectively as possible with Scrum values. This means they keep the team on track, plan and lead meetings, and work out any obstacles the team might face. Scrum Masters might also work in a larger role within an organization to help it incorporate Scrum concepts into their work. Because they are both a leader and a behind-the-scenes supporter, they are often described as the “servant leader” of the Scrum team.

Scrum can look different from organization to organization and team to team, making the specific tasks of a Scrum Master varied. Broadly, however, a Scrum Master might have the following responsibilities:

* Facilitate daily Scrum meetings (also called “daily standups”)
* Lead sprint planning meetings
* Conduct “retrospective” reviews to see what went well and what can be improved for the following sprint
* Keep a pulse on team members, through individual meetings or other means of communication
* Manage obstacles that arise for the team by communicating with stakeholders outside of the team

**Read more:** [7 In-Demand Scrum Master Certifications](https://www.coursera.org/articles/scrum-master-certifications)

## 2. Product owner

A product owner makes sure the Scrum team is aligned with the goals of the overall product that the team is contributing to. They understand the business needs of the product, like customer expectations, and market trends. Because they have to understand how the Scrum team fits into bigger picture goals, product owners usually stay in touch with product managers and other stakeholders outside the team.

Product owners might find themselves with the following responsibilities:

* Manage the product backlog by ordering work by priority
* Set the product vision for the team
* Communicate with external stakeholders and translate their needs to the team
* Make sure the team is focused on hitting product needs through communication and evaluating progress

## 3. Development team

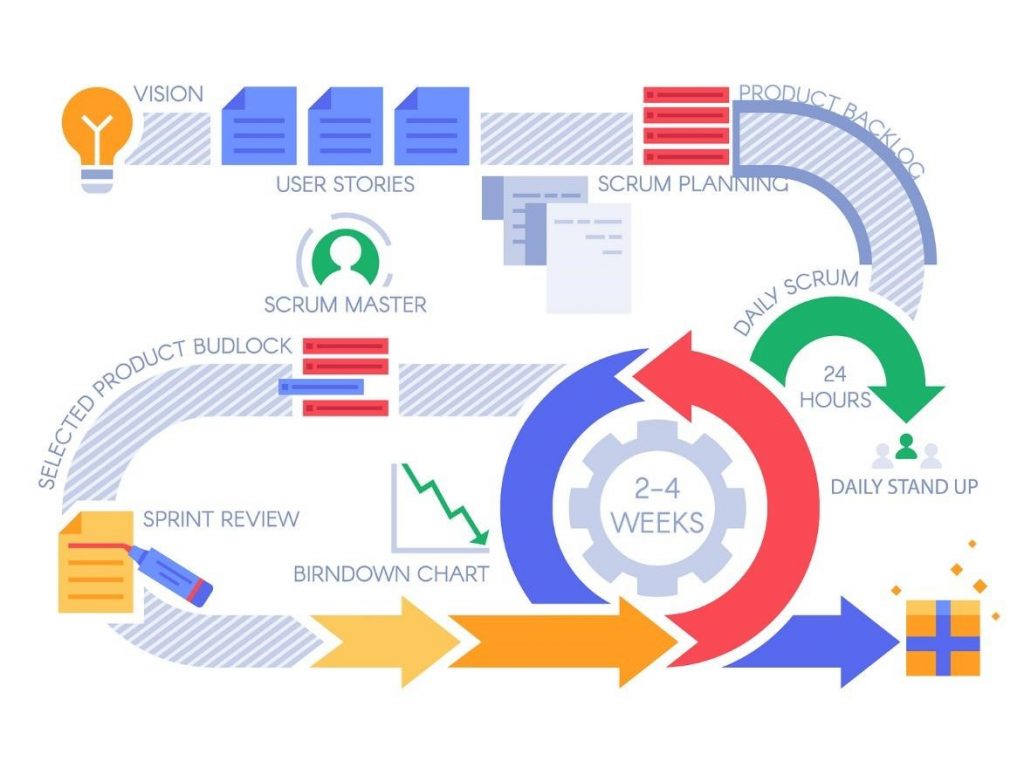
A development team is composed of professionals who do the hands-on work of completing the tasks in a Scrum sprint. This means development team members can be computer engineers, designers, writers, data analysts, or any other role needed to reach sprint goals. The development team doesn’t just wait for orders; they usually collaborate to map out goals and plans for achieving them.

Not all development team members will always have the same responsibilities. For example, if you’re updating a website, you might have a front-end engineer, UX designer, copywriter, and marketing professional all working on the same Scrum team. The responsibilities of a development team will also depend on the end goals of the Scrum team. Broadly speaking, however, you might find that a development team can be tasked with the following:

* Help in sprint planning and goal setting
* Lend expertise to program, design, or improve products
* Use data to find best practices for development
* Test products and prototypes, plus other forms of quality assurance

# What is Scrum?

Scrum is an [agile development methodology](https://www.digite.com/agile/agile-methodology/) used in the development of Software based on an iterative and incremental processes.  Scrum is adaptable, fast, flexible and effective agile framework that is designed to deliver value to the customer throughout the development of the project. The primary objective of Scrum is to satisfy the customer’s need through an environment of transparency in communication, collective responsibility and continuous progress. The development starts from a general idea of ​​what needs to be built, elaborating a list of characteristics ordered by priority (product backlog) that the owner of the product wants to obtain.



##### Learn Scrum:

##### **A brief History on Scrum**

The history of Scrum can be traced back to 1986 in the Harvard Business Review (HBR) article titled, “[The New Product Development Game](https://hbr.org/1986/01/the-new-new-product-development-game)” by Hirotaka Takeuchi & Ikujiro Nonaka. This article describes how companies such as Honda, Canon, and Fuji-Xerox produce new products worldwide using a scalable and team-based approach to product development. This approach emphasizes the importance of empowering self-organized teams.



The article was an influence to develop many of the concepts that gave birth to what we now call Scrum. Scrum is a term drawn from Rugby, which refers to how the game is restarted after a foul or when the ball has left the game.

In 1993, [Jeff Sutherland](https://en.wikipedia.org/wiki/Jeff_Sutherland) and his team at Easel Corporation created the Scrum process to be used in software development processes by combining the concepts of the 1986 article with the concepts of object-oriented development, empirical process control, iterative development and incremental, software processes and productivity improvement, as well as the development of complex and dynamic systems.

##### **Scrum Methodology & Process**

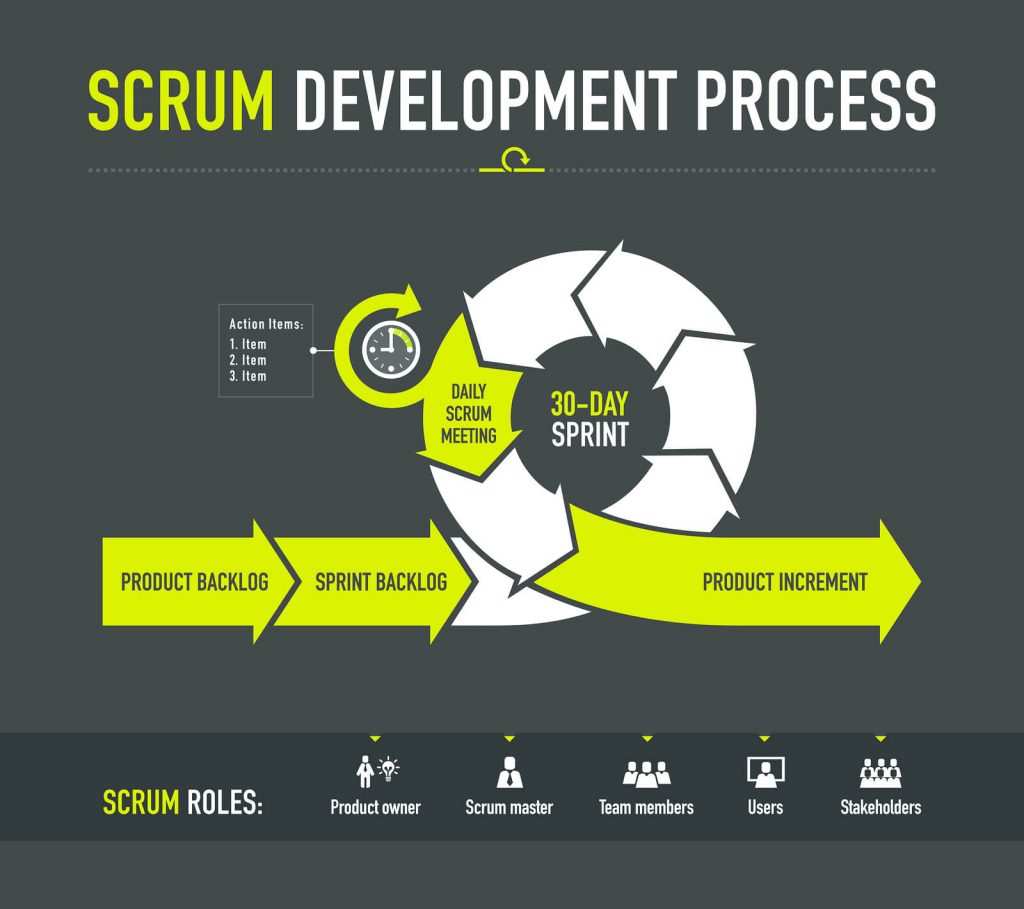
Scrum is precisely an evolution of Agile Management. Scrum methodology is based on a set of very defined practices and roles that must be involved during the software development process. It is a flexible methodology that rewards the application of the [12 agile principles](https://www.digite.com/agile/agile-methodology/#agile-principles) in a context agreed by all the team members of the product.

Scrum is executed in temporary blocks that are short and periodic, called Sprints, which usually range from 2 to 4 weeks, which is the term for feedback and reflection. Each Sprint is an entity in itself, that is, it provides a complete result, a variation of the final product that must be able to be delivered to the client with the least possible effort when requested.

The process has as a starting point, a list of objectives/ requirements that make up the project plan. It is the client of the project that prioritizes these objectives considering a balance of the value and the cost thereof, that is how the iterations and consequent deliveries are determined.

On the one hand the market demands quality, fast delivery at lower costs, for which a company must be very agile and flexible in the development of products, to achieve short development cycles that can meet the demand of customers without undermining the quality of the result. It is a very easy methodology to implement and very popular for the quick results it gets.

Scrum methodology is used mainly for software development, but [**other sectors**](https://www.benlinders.com/2016/agile-scrum-beyond-sw-development/) are also taking advantage of its benefits by implementing this methodology in their organizational models such as sales, marketing, & HR teams etc.



##### **Different Roles in Scrum**

In Scrum, the team focuses on building quality software. The owner of a Scrum project focuses on defining what are the characteristics that the product must have to build (what to build, what not and in what order) and to overcome any obstacle that could hinder the task of the development team.

**The Scrum team consists of the following roles:**

**Scrum master:** The person who leads the team guiding them to comply with the rules and processes of the methodology. Scrum master manages the reduction of impediments of the project and works with the Product Owner to maximize the ROI. The Scrum Master is in charge of keeping Scrum up to date, providing coaching, mentoring and training to the teams in case it needs it.

**Product owner (PO):** Is the representative of the stakeholders and customers who use the software. They focus on the business part and is responsible for the ROI of the project. They Translate the vision of the project to the team, validate the benefits in stories to be incorporated into the Product Backlog and prioritize them on a regular basis.

**Team:** A group of professionals with the necessary technical knowledge who develop the project jointly carrying out the stories they commit to at the start of each sprint.



##### **Benefits of Scrum Methodology**

Scrum has many advantages over other agile development methodologies. It is currently the most used and trusted framework of reference in the software industry. Below are some of the known benefits of Scrum:

**Easily Scalable:** Scrum processes are iterative and are handled within specific work periods, which makes it easier for the team to focus on definite functionalities for each period. This not only has the benefit of achieving better deliverables in line with the needs of the user, but also gives the ability to the teams to scale the modules in terms of functionality, design, scope and characteristics in an orderly, transparent and simple manner.

**Compliance of expectations:** The client establishes their expectations indicating the value that each requirement/ history of the project brings, the team estimates them and with this information the Product Owner establishes its priority. On a regular basis, in the sprint demos, the Product Owner verifies that the requirements have been met and transmits feedback to the team.

**Flexible to changes:** Quick reaction to changes in requirements generated by customer needs or market developments. The methodology is designed to adapt to the changing requirements that complex projects entail.

**Time to Market reduction:** The client can start using the most important functionalities of the project before the product is completely ready.

**Higher software quality:** The working method and the need to obtain a functional version after each iteration, helps to obtain a higher quality software.

**Timely Prediction:**  Using this methodology, we know the average speed of the team by sprint (story points), with which, consequently, it is possible to estimate when a certain functionality that is still in the backlog will be available.

**Reduction of risks:**  The fact of carrying out the most valuable functionalities in the first place and of knowing the speed with which the team advances in the project, allows to clear risks effectively in advance.

##### **Events in Scrum**

Each of the Scrum events facilitates the adaptation of some of the aspects of the process, the product, progress or relationships.

**Sprint:** Sprint is the basic unit of work for a Scrum team. This is the main feature that marks the difference between Scrum and other models for agile development.

**Sprint Planning:** The goal of the Sprint Planning is to define what is going to be done in the Sprint and how it is going to be done. This meeting is held at the beginning of each Sprint and is defined how it will approach the project coming from the Product Backlog stages and deadlines. Each Sprint is composed of different features.

**Daily Scrum:** The objective of the Daily Scrum is to evaluate the progress and trend until the end of the Sprint, synchronizing the activities and creating a plan for the next 24 hours. It is a brief meeting that takes place daily during the Sprint period. Three questions are answered individually:  What did I do yesterday? What am I going to do today? What help do I need?  The Scrum Master should try to solve problems or obstacles that arise.

**Sprint Review:** The goal of the sprint review is to show what work has been completed with regards to the product backlog for future deliveries. The finished sprint is reviewed, and there should already be a clear and tangible advancement in the product to present to the client.

**Sprint Retrospective:** The team reviews the completed goals of the finished sprint, write down the good and the bad, so as not to repeat the mistakes again. This stage serves to implement improvements from the point of view of the development process. The goal of the sprint retrospective is to identify possible process improvements and generate a plan to implement them in the next Sprint.

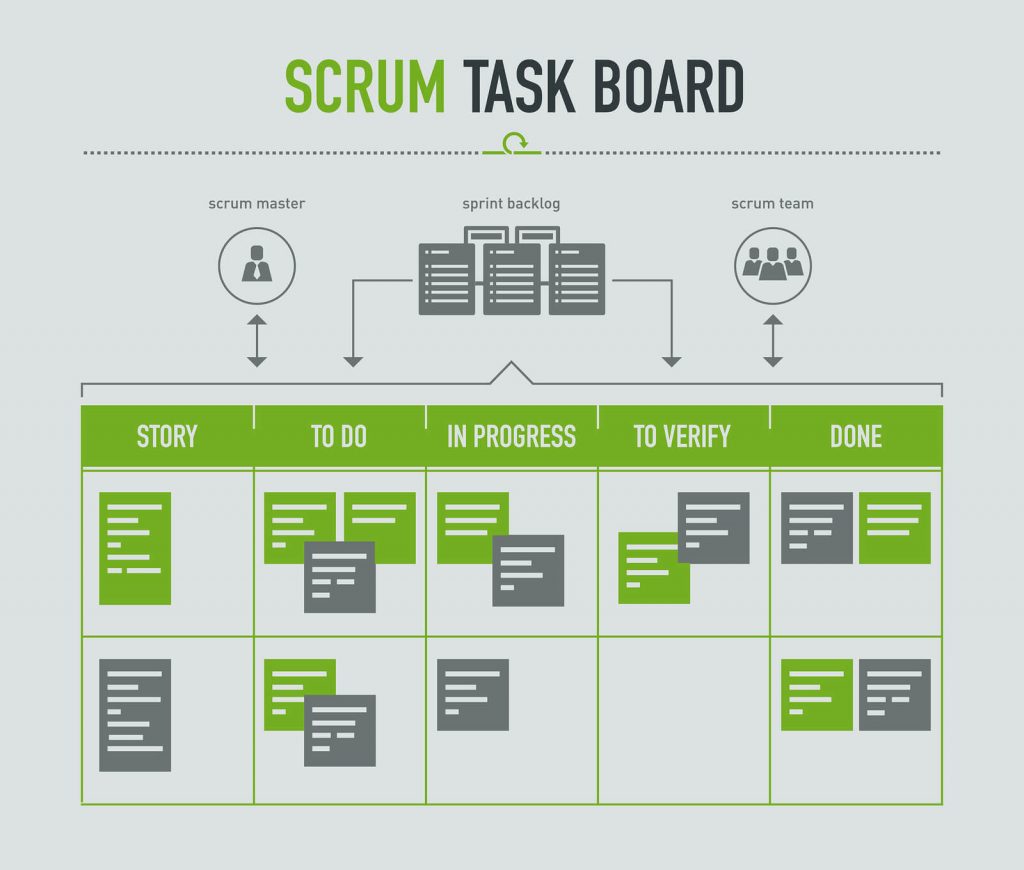
##### **Scrum Artifacts**

Scrum Artifacts are designed to guarantee the transparency of key information in decision making.

**Product Backlog (PB):** The product backlog is a list that collects everything the product needs to satisfy the potential customers. It is prepared by the product owner and the functions are prioritized according to what is more and less important for the business. The goal is for the product owner to answer the question “What should be done”.

**Sprint Backlog (SB):** It is a subset of items of the product backlog, which are selected by the team to perform during the sprint on which they are going to work. The team establishes the duration of each Sprint. Usually the sprint backlog, is displayed on physical boards called as Scrum board – that makes the development process visible to everyone who enters the development area.

**Increment:** The Increment is the sum of all the tasks, use cases, user stories, product backlogs and any element that was developed during the sprint and that will be made available to the end user in the form of Software.



##### **Planning in Scrum**

The Sprint Planning Meeting is held at the beginning of each Sprint. All the members of the Team participate in the meeting, i.e., the Product Owner, Scrum Master and all the Development Team. The entire Scrum team must understand and define what objective should be obtained in that Sprint (Sprint Goal). From this point the development team must design a work plan to achieve the objective. This planning should allow you to see if the sprint goal involves a workload according to the duration stipulated for the Sprints (which is 2 to 4 weeks).

The client shows the result to be achieved in that Sprint and the requirements of the deliverable product. Here you have to carry out a discussion in which the development team evaluates what elements of the list can be delivered.

Both the Scrum Master and the Product Owner must collaborate to clarify any aspect of the requirements. Finally, the development team must explain how they will organize the team’s work to achieve the Sprint goal.

## What Is DevOps?

DevOps is a set of [practices](https://www.atlassian.com/devops/what-is-devops/devops-best-practices), [tools](https://www.atlassian.com/devops/devops-tools/choose-devops-tools), and a [cultural philosophy](https://www.atlassian.com/devops/what-is-devops/devops-culture) that automate and integrate the processes between software development and IT teams. It emphasizes team empowerment, cross-team communication and collaboration, and technology automation.

The DevOps movement [began around 2007](https://www.atlassian.com/devops/what-is-devops/history-of-devops) when the software development and IT operations communities raised concerns about the traditional software development model, where developers who wrote code worked apart from operations who deployed and supported the code. The term DevOps, a combination of the words development and operations, reflects the process of integrating these disciplines into one, continuous process.

## How does DevOps work?

A DevOps team includes developers and IT operations working collaboratively throughout the product lifecycle, in order to increase the speed and quality of software deployment. It’s a new way of working, a cultural shift, that has significant implications for teams and the organizations they work for.

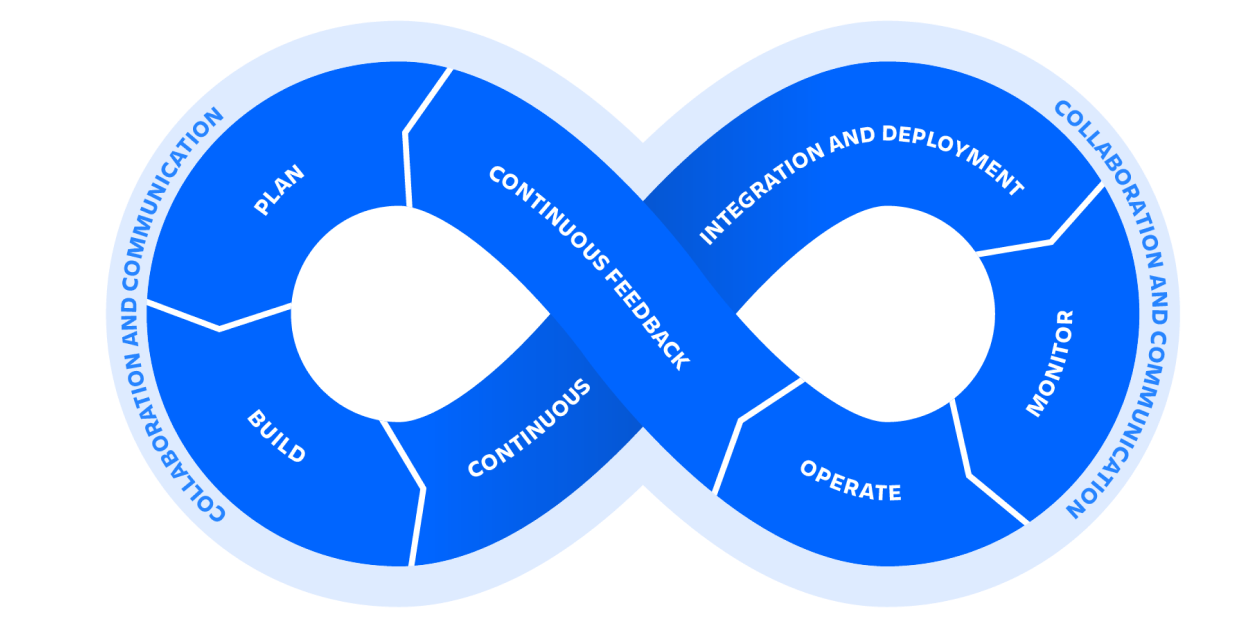
Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams merge into a single team where the engineers work across the entire application lifecycle — from development and test to deployment and operations — and have a range of multidisciplinary skills.

DevOps teams use tools to automate and accelerate processes, which helps to increase reliability. A DevOps toolchain helps teams tackle important DevOps fundamentals including continuous integration, continuous delivery, automation, and collaboration.

DevOps values are sometimes applied to teams other than development. When security teams adopt a DevOps approach, security is an active and integrated part of the development process. This is called [DevSecOps](https://www.atlassian.com/continuous-delivery/principles/devsecops).

## The DevOps lifecycle

Because of the continuous nature of DevOps, practitioners use the infinity loop to show how the phases of the DevOps lifecycle relate to each other. Despite appearing to flow sequentially, the loop symbolizes the need for constant collaboration and iterative improvement throughout the entire lifecycle.



The DevOps lifecycle consists of six phases representing the processes, capabilities, and tools needed for development (on the left side of the loop) and operations (on the right side of the loop). Throughout each phase, teams collaborate and communicate to maintain alignment, velocity, and quality.

[Learn more about communication and collaboration](https://www.atlassian.com/work-management)

### Plan

DevOps teams should adopt agile practices to improve speed and quality. Agile is an iterative approach to project management and software development that helps teams break work into smaller pieces to deliver incremental value.

[Learn more about agile](https://www.atlassian.com/agile)

### Build

Git is a free and open source version control system. It offers excellent support for branching, merging, and rewriting repository history, which has led to many innovative and powerful workflows and tools for the development build process.

[Learn more about Git](https://www.atlassian.com/git)

### Continuous integration and delivery

CI/CD allows teams to release quality products frequently and predictably, from source code repository to production with automated workflows. Teams can merge code changes frequently, deploy feature flags, and incorporate end-to-end testing.

[Learn more about CI/CD](https://www.atlassian.com/continuous-delivery)

### Monitor and alert

Quickly identify and resolve issues that impact product uptime, speed, and functionality. Automatically notify your team of changes, high-risk actions, or failures, so you can keep services on.

[Learn more about monitoring](https://www.atlassian.com/incident-management/incident-response)

### Operate

Manage the end-to-end delivery of IT services to customers. This includes the practices involved in design, implementation, configuration, deployment, and maintenance of all IT infrastructure that supports an organization’s services.

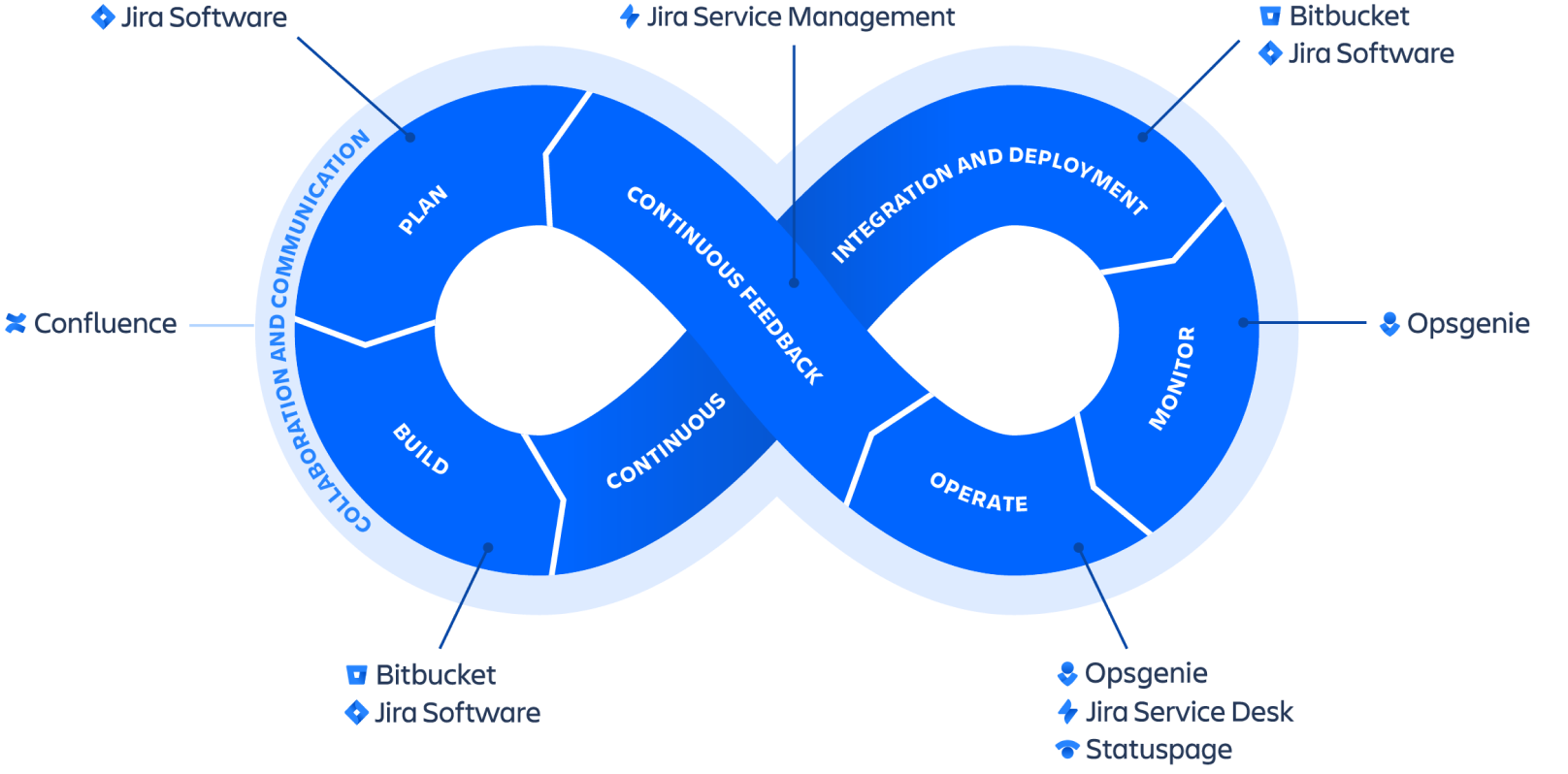
[Learn more about operations](https://www.atlassian.com/itsm)

### Continuous feedback

DevOps teams should evaluate each release and generate reports to improve future releases. By gathering continuous feedback, teams can improve their processes and incorporate customer feedback to improve the next release.

[Learn more about continuous feedback](https://www.atlassian.com/software/jira/service-management/resources)

## DevOps tools



[DevOps tools](https://www.atlassian.com/devops/devops-tools) address the key phases of the DevOps lifecycle. They empower DevOps practices by helping to improve collaboration, reduce context-switching, introduce automation, and enable observability and monitoring.

DevOps toolchains usually follow two approaches: an all-in-one or open toolchain. An all-in-one toolchain offers a complete solution that usually doesn’t integrate with other third-party tools, while an open toolchain allows for customization with different tools. There are [pros and cons to both approaches.](https://www.atlassian.com/devops/devops-tools/choose-devops-tools)

An example of an open DevOps toolchain is Atlassian’s Open DevOps solution, which includes Jira as a foundation and integrates with leading vendors and marketplace apps.

[Read more about DevOps tools](https://www.atlassian.com/devops/devops-tools)

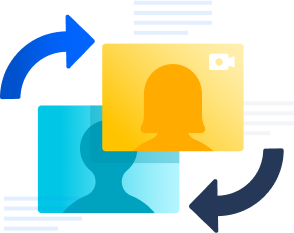
## What are the benefits of DevOps?

In Atlassian’s [2020 DevOps Trends survey](https://www.atlassian.com/whitepapers/devops-survey-2020), 99 percent of respondents said that DevOps had a positive impact on their organization. The [benefits of DevOps](https://www.atlassian.com/devops/what-is-devops/benefits-of-devops) include faster and easier releases, team efficiency, increased security, higher quality products, and consequently happier teams and customers.



### Speed

Teams that practice DevOps release deliverables more frequently, with higher quality and stability. In fact, the DORA [2019 State of DevOps](https://cloud.google.com/blog/products/devops-sre/the-2019-accelerate-state-of-devops-elite-performance-productivity-and-scaling) report found that elite teams deploy 208 times more frequently and 106 times faster than low-performing teams. Continuous delivery allows teams to build, test, and deliver software with automated tools.



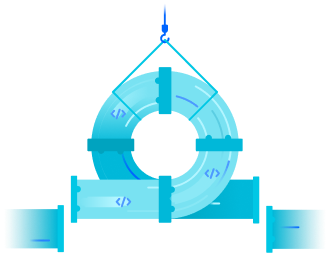
### Improved collaboration

The foundation of DevOps is a culture of collaboration between developers and operations teams, who share responsibilities and combine work. This makes teams more efficient and saves time related to work handoffs and creating code that is designed for the environment where it runs.



### Rapid deployment

By increasing the frequency and velocity of releases, DevOps teams improve products rapidly. A competitive advantage can be gained by quickly releasing new features and repairing bugs.



### Quality and reliability

Practices like continuous integration and continuous delivery ensure changes are functional and safe, which improves the quality of a software product. Monitoring helps teams keep informed of performance in real-time.



### Security

By integrating security into a continuous integration, continuous delivery, and continuous deployment pipeline, [DevSecOps](https://www.atlassian.com/devops/devops-tools/devsecops-tools) is an active, integrated part of the development process. Security is built into the product by integrating active security audits and security testing into agile development and DevOps workflows.

## What are the challenges of adopting DevOps?

Habits are hard to break. Teams entrenched in siloed ways of working can struggle with, or even be resistant to, overhauling team structures to embrace DevOps practices. Some teams may mistakenly believe new tools are sufficient to adopt DevOps. Yet, DevOps is a combination of people, tools, and culture. Everyone on a DevOps team must understand the entire value stream — from ideation, to development, to the end user experience. It requires breaking down silos in order to collaborate throughout the product lifecycle.

## Devops isn't any single person's job. It's everyone's job.

Robert Krohn

##### Head of Engineering, DevOps at Atlassian

Moving from a legacy infrastructure to using [Infrastructure as Code](https://www.atlassian.com/microservices/cloud-computing/infrastructure-as-code) (IaC) and [microservices](https://www.atlassian.com/continuous-delivery/microservices) can offer faster development and innovation, but the increased operational workload can be challenging. It’s best to build out a strong foundation of automation, configuration management, and continuous delivery practices to help ease the load.

An over-reliance on tools can detract teams from the necessary foundations of DevOps: the team and organization structure. Once a structure is established, the processes and team should come next and the tools should follow.

## How to adopt DevOps

Adopting DevOps first requires a commitment to evaluating and possibly changing or removing any teams, tools, or processes your organization currently uses. It means building the necessary infrastructure to give teams the autonomy to build, deploy, and manage their products without having to rely too heavily on external teams.

### DevOps culture

A [DevOps culture](https://www.atlassian.com/devops/what-is-devops/devops-culture) is where teams embrace new ways of working that involve greater collaboration and communication. It’s an alignment of people, processes, and tools toward a more unified customer focus. Multidisciplinary teams take accountability for the entire lifecycle of a product.

### Continuous learning

Organizations that do DevOps well are places where experimentation and some amount of risk-taking are encouraged. Where thinking outside the box is the norm, and failure is understood to be a natural part of learning and improving.

### Agile

[Agile methodologies](https://www.atlassian.com/agile) are immensely popular in the software industry since they empower teams to be inherently flexible, well-organized, and capable of responding to change. DevOps is a cultural shift that fosters collaboration between those who build and maintain software. When used together, agile and DevOps result in high efficiency and reliability.

## DevOps practices

### Continuous integration

[Continuous integration](https://www.atlassian.com/continuous-delivery/continuous-integration) is the practice of automating the integration of code changes into a software project. It allows developers to frequently merge code changes into a central repository where builds and tests are executed. This helps DevOps teams address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

[Learn about continuous integration](https://www.atlassian.com/continuous-delivery/continuous-integration)

### Continuous delivery

Continuous delivery expands upon continuous integration by automatically deploying code changes to a testing/production environment. It follows a [continuous delivery pipeline](https://www.atlassian.com/continuous-delivery/pipeline), where automated builds, tests, and deployments are orchestrated as one release workflow.

[Learn about continuous delivery](https://www.atlassian.com/continuous-delivery)

### Situational awareness

It is vital for every member of the organization to have access to the data they need to do their job as effectively and quickly as possible. Team members need to be alerted of failures in the deployment pipeline — whether systemic or due to failed tests — and receive timely updates on the health and performance of applications running in production. Metrics, logs, traces, monitoring, and alerts are all essential sources of feedback teams need to inform their work.

[Learn about DevOps metrics](https://www.atlassian.com/devops/frameworks/devops-metrics)

### Automation

Automation is one of the most important DevOps practices because it enables teams to move much more quickly through the process of developing and deploying high-quality software. With automation the simple act of pushing code changes to a source code repository can trigger a build, test, and deployment process that significantly reduces the time these steps take.

[Learn about DevOps automation best practices](https://www.atlassian.com/solutions/devops/guides/intermediate)

### Infrastructure as Code

Whether your organization has an on-premise data center or is completely in the cloud, having the ability to quickly and consistently provision, configure, and manage infrastructure is key to successful DevOps adoption. [Infrastructure as Code](https://www.atlassian.com/continuous-delivery/principles/infrastructure-as-code) (IaC) goes beyond simply scripting infrastructure configuration to treating your infrastructure definitions as actual code: using source control, code reviews, tests, etc.

[Learn about Infrastructure as Code](https://www.atlassian.com/microservices/cloud-computing/infrastructure-as-code)

### Microservices

[Microservices](https://www.atlassian.com/continuous-delivery/microservices) is an architectural technique where an application is built as a collection of smaller services that can be deployed and operated independently from each other. Each service has its own processes and communicates with other services through an interface. This separation of concerns and decoupled independent function allows for DevOps practices like continuous delivery and continuous integration.

[Learn about microservices](https://www.atlassian.com/continuous-delivery/microservices)

### Monitoring

DevOps teams monitor the entire development lifecycle — from planning, development, integration and testing, deployment, and operations. This allows teams to respond to any degradation in the customer experience, quickly and automatically. More importantly, it allows teams to “shift left” to earlier stages in development and minimize broken production changes.