**What is Waterfall Model ?**

The waterfall model is a linear, sequential approach to the [software development lifecycle](https://www.techtarget.com/searchsoftwarequality/definition/systems-development-life-cycle) (SDLC) that is popular in software engineering and [product development](https://www.techtarget.com/searchcio/definition/product-development-or-new-product-development-NPD).

The waterfall model uses a logical progression of SDLC steps for a project, similar to the direction water flows over the edge of a cliff. It sets distinct endpoints or goals for each phase of development. Those endpoints or goals can't be revisited after their completion.

Dr. Winston W. Royce at the Lockheed Software Technology Center introduced the concept in a paper published in 1970 on his experience developing software for satellites. However, Royce didn't use the term waterfall; instead, he referred to the downstream value of documentation.

The waterfall model continues to be used in industrial design applications. It's often cited as the first software development methodology. The model is also used more generally as a high-level project management methodology for complicated, multifaceted projects.

**Phases of the waterfall model**

When used for a software development process, the waterfall methodology has seven stages:

1. **Requirements**. Potential requirements, deadlines and guidelines for the project are analyzed and placed into a formal requirements document, also called a [*functional specification*](https://www.techtarget.com/searchsoftwarequality/definition/functional-specification). This stage of development defines and plans the project without mentioning specific processes.
2. **Analysis.** The system specifications are analyzed to generate product models and [business logic](https://www.techtarget.com/whatis/definition/business-logic) to guide production. This is also when financial and technical resources are audited for feasibility.
3. **Design.** A [design specification document](https://www.techtarget.com/searchsoftwarequality/tip/A-guide-to-software-design-documentation-and-specifications) is created to outline technical design requirements, such as the programming language, [hardware](https://www.techtarget.com/searchnetworking/definition/hardware), data sources, architecture and services.
4. **Coding and implementation.** The [source code](https://www.techtarget.com/searchapparchitecture/definition/source-code) is developed using the models, logic and requirement specifications designated in the prior phases. Typically, the system is coded in smaller components, or units, before being put together.
5. **Testing.** This is when [quality assurance](https://www.techtarget.com/searchsoftwarequality/definition/quality-assurance), [unit](https://www.techtarget.com/searchsoftwarequality/definition/unit-testing), [system](https://www.techtarget.com/searchsoftwarequality/definition/system-testing) and [beta](https://www.techtarget.com/whatis/definition/beta-test) tests identify issues that must be resolved. This may cause a forced repeat of the coding stage for [debugging](https://www.techtarget.com/searchsoftwarequality/definition/debugging). If the system passes integration and testing, the waterfall continues forward.
6. **Operation and deployment.** The product or application is deemed fully functional and is deployed to a live environment.
7. **Maintenance.** Corrective, adaptive and perfective maintenance is carried out indefinitely to improve, update and enhance the product and its functionality. This could include releasing [patch](https://www.techtarget.com/searchenterprisedesktop/definition/patch) updates and new versions.

**Advantages of Waterfall Model ?**

Today, Agile methodology is often [used in place](https://www.pmi.org/learning/library/agile-versus-waterfall-approach-erp-project-6300) of the waterfall model. However, there are advantages to the waterfall approach, such as the following:

* enables large or changing teams to move toward a common goal that's been defined in the requirements stage;
* forces structured, disciplined organization;
* simplifies understanding, following and arranging tasks;
* facilitates departmentalization and managerial control based on the schedule or deadlines;
* reinforces [good coding habits](https://www.techtarget.com/searchsoftwarequality/feature/Learn-5-defensive-programming-techniques-from-experts) to define before implementing design and then code;
* enables early system design and specification changes to be easily done; and
* clearly defines milestones and deadlines.

**Disadvantages of the waterfall model**

Disadvantages of the waterfall model typically center around the risk associated with a lack of revision and flexibility. Specific issues include the following:

* Design isn't adaptive; when a flaw is found, the entire process often needs to start over.
* Method doesn't incorporate midprocess user or client feedback, and makes changes based on results.
* Waterfall model delays testing until the end of the development lifecycle.
* It doesn't consider error correction.
* The methodology doesn't handle requests for changes, [scope](https://www.techtarget.com/searchcio/definition/project-scope) adjustments and updates well.
* Waterfall doesn't let processes overlap for simultaneous work on different phases, reducing overall efficiency.
* No working product is available until the later stages of the project lifecycle.
* Waterfall isn't ideal for complex, high-risk ongoing projects.

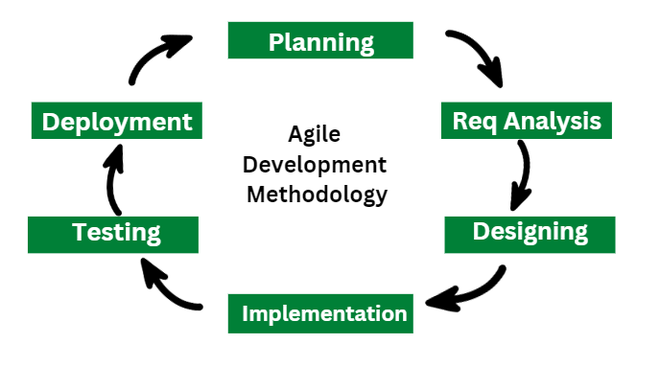
**What is Agile Methodogy ?**

Agile methodologies are iterative and incremental, which means it’s known for breaking a project into smaller parts and adjusting to changing requirements.

1. They prioritize flexibility, collaboration, and customer satisfaction.
2. Major companies like Facebook, Google, and Amazon use Agile because of its adaptability and customer-focused approach.

## Life cycle of Agile Methodology

The [Agile software development life cycle](https://www.geeksforgeeks.org/agile-sdlc-software-development-life-cycle/) helps you break down each project you take on into six simple stages:



**Advantages of Agile Methodology**

1. **Immediate Feedback:**It allows immediate feedback, which aids software improvement in the next increment.
2. **Adapts to Changing Requirements:** It is a highly adaptable methodology in which rapidly changing requirements, allowing responsive adjustments.
3. **Face-to-Face Communication:** Agile methodology encourages effective face-to-face communication.
4. **Time-Efficient:**It is well-suited for its time-efficient practices, which help in delivering software quickly and reducing time-to-market.
5. **Frequent Changes:**It effectively manages and accommodates frequent changes in project requirements according to stakeholder convenience.
6. **Customer Satisfaction:** It prioritizes customer satisfaction.
7. **Flexibility and Adaptability:**Agile methodologies are known for their flexibility and adaptability.

**Disadvantages of the Agile Methodology**

Following are the disadvantages of the agile methodology:

1. **Lack of Predictability:** Project timeframes and outcomes might be difficult to predict with accuracy due to Agile iterative and incremental methodology. Stakeholders who need set budgets or timeframes may find this unpredictability troublesome.
2. **Dependency on Customer Availability:** Agile highly depends on ongoing customer and stakeholder feedback and participation. Customers who are unavailable or who don’t know enough about the domain can impede development and slow it down.
3. **Scaling Agile**: While Agile works effectively for small to medium-sized teams working on relatively basic projects, scaling Agile methods to bigger teams or more complicated projects can be more difficult. As the project grows, it gets harder to maintain coordination, alignment, and communication.
4. **Dependency on Team Dynamics:** Agile’s focus on self-organizing, cross-functional teams with the authority to reach decisions together is paramount. Inadequate communication within the team or a lack of experience or expertise among team members can negatively affect output quality and productivity.
5. **Increased Overhead:** Planning, coordinating, and communicating take more time and effort when using agile frameworks like Scrum. This overhead can take a lot of time, especially for projects with short deadlines or small teams.

**Scrum Master**

A Scrum Master is a professional holding a position with a significantly narrow scope but a broad range of influence in any organisation. However, practically speaking, a Scrum Master works at the backend and does not involve himself in the strategy or product ideation processes. They work as catalysts between product owners or business owners and the development team, similar to a project manager. With agile processes almost completely dependent on collaboration and manual effort, Scrum Masters need to perfectly blend the most advanced processes and tools with high-level soft skills.

A software project comes with multiple moving components, so individual programmers can lose sight of their broader scope while focusing only on the code. Meanwhile, a Scrum Master maintains a big-picture perspective, simultaneously helping teams to understand all technical and organisational dependencies and avoiding any bottlenecks. This leads to a culture that fosters accountability and sticks to deadlines.

**Product Owner**

The Product Owner (PO) is the Agile team member primarily responsible for maximizing the value delivered by the team by ensuring that the team backlog is aligned with customer and stakeholder needs. As a member of the extended Product Management function, the PO is the team’s primary customer advocate and primary link to business and technology strategy. This enables the team to balance the needs of multiple stakeholders while continuously evolving the Solution.

**What is CI / CD ?**

CI and CD is the practice of automating the integration of code changes from multiple developers into a single codebase. It is a software development practice where the developers commit their work frequently to the central code repository (Github or Stash). Then there are automated tools that build the newly committed code and do a code review, etc as required upon integration.

The key goals of Continuous Integration are to find and address bugs quicker, make the process of integrating code across a team of developers easier, improve software quality, and reduce the time it takes to release new feature updates. Some popular CI tools are Jenkins, TeamCity, and Bamboo.

**Advantages of CI/CD:**

* **Improved collaboration**

In a CI/CD pipeline, all roles work together, which improves communication and collaboration.

* **Better code quality**

CI/CD can help improve the overall quality and security of the software.

* **Reduced risk**

Testing and deploying code more frequently can reduce the risk of bugs and code defects, making them easier and cheaper to fix.

* **Faster time to repair**

CI/CD can reduce the average time to repair a broken feature, or MTTR, because code changes are smaller and fault isolations are easier to detect.

* **Cost deduction**

A CI/CD pipeline can limit the potential impact of a deployment problem by allowing it to be deployed in non-critical business hours.

**Disadvantages to CI/CD:**

* **Requires businesses to be alert and iterative**

Businesses need to be alert and iterative to avoid choosing the wrong automation process and to be extra cautious in picking the right order of process.

* **Code base needs to be ready**

The code base needs to be ready and immediately put to production once the current result is successful.

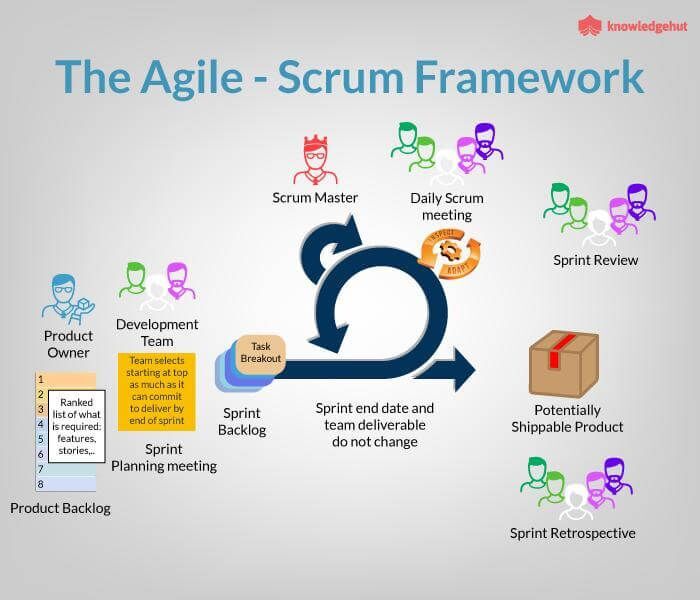
* **Teams may make dashboards that not every member knows**

If teams make dashboards that not every member knows beforehand, they may fall prey to logical fallacy.

* **CI and CD need to be implemented in sync**

CI and CD need to be implemented in sync with each other, which requires a lot of attention and detailing with respect to the human factor

**Scrum Framework**



In Scrum, the prescribed events are used to create regularity. All events are time-boxed events, such that every event has a maximum duration. The events are described more elaborately in the subsequent chapters.

## Sprint

The heart of Scrum is a Sprint, a time-box of two weeks or one month during which a potentially releasable product increment is created. A new Sprint starts immediately after the conclusion of the previous Sprint. Sprints consist of the Sprint planning, daily scrums, the development work, the Sprint review, and the Sprint retrospective.

* In Sprint planning, the work to be performed in the Sprint is planned collaboratively by the Scrum Team.
* The Daily Scrum Meeting is a 15-minute time-boxed event for the Scrum Team to synchronize the activities and create a plan for that day.
* A Sprint Review is held at the end of the Sprint to inspect the Increment and make changes to the Product Backlog, if needed.
* The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. In this meeting, the Scrum Team is to inspect itself and create a plan for improvements to be enacted during the subsequent Sprint.

**Advantage of Scrum framework**

* Scrum framework is fast moving and money efficient.
* Scrum framework works by dividing the large product into small sub-products. It’s like a divide and conquer strategy
* In Scrum customer satisfaction is very important.
* Scrum is adaptive in nature because it have short sprint.
* As Scrum framework rely on constant feedback therefore the quality of product increases in less amount of time

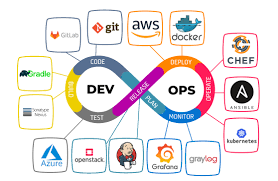
**Disadvantage of Scrum framework**

* Scrum framework do not allow changes into their sprint.
* Scrum framework is not fully described model. If you wanna adopt it you need to fill in the framework with your own details like [Extreme Programming(XP),](https://www.geeksforgeeks.org/software-engineering-extreme-programming-xp/) [Kanban](https://www.geeksforgeeks.org/kanban-agile-methodology/), [Dynamic Systems Development Method (DSDM)](https://www.geeksforgeeks.org/dynamic-systems-development-method-dsdm/).
* It can be difficult for the Scrum to plan, structure and organize a project that lacks a clear definition.
* The daily Scrum meetings and frequent reviews require substantial resources.

## 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.



### **Advantages of DevOps**

* Faster development and deployment of applications.
* Faster response to the market changes to improve business growth.
* Business profit is increased as there is a decrease in software delivery time and transportation costs.
* Improves customer experience and satisfaction.
* Simplifies collaboration as all the tools are placed in the cloud for customers to access.
* Leads to better team engagement and productivity due to collective responsibility.

### **Disadvantages of DevOps**

* Less availability of DevOps professionals.
* Infrastructure cost is high for setting by DevOps environment.
* Lack of DevOps knowledge can lead to problems in the continuous integration of automation projects.

**Find all git commands.**

### 1. git help

Take help from the Git help section for different commands and other errors.

git help

### 2. git config

To set the basic configurations on Git like your name and email.

git config

### 3. git config –-global user.name “ ”

Sets configuration values for your user name on git.

git config –-global user.name “Ashish Madaan”

### 4. git config –-global user.email ” “

Sets configuration values for your user email on git.

git config –-global user.email ashishmadaan6@gmail.com

### 5. git config –-global color.ui

To see different colors on the command line for different outputs.

git config –-global color.ui true

### 6. mkdir

Create a directory if not created initially.

mkdir store

### 7. cd

To go inside the directory and work on its contents.

cd store

### 8. git init

To create a local git repository for us in our store folder. This will help to manage the git commands for that particular repository.

git init

### 9. git status

To see what’s changed since the last commit. It shows all the files that have been added and modified and are ready to be committed and files that are untracked.

git status

### 10. git add Readme.txt

To add a file Readme.txt to the staging area to track its changes.

git add Readme.txt

### 11. git commit -m “ ”

To commit our changes(taking a snapshot) and provide a message to remember for future reference.

git commit -m “Created a Readme.txt”

### 12. git log

To check the history of commits for our reference.

git log

### 13. git add

To add a specific list of files to the staging area.

git add

### 14. git add –all

To add all files of the current directory to the staging area.

git add --all

### 15. git add \*.txt

To add all text files of the current directory to the staging area.

git add \*.txt

### 16. git add docs/\*.txt

To add all text files of a particular directory(docs) to the staging area.

git add docs/\*.txt

### 17. git add docs/

To add all files in a particular directory(docs) to the staging area.

git add docs/

### 18. git add “\*.txt”

To add text files of the entire project to the staging area.

git add “\*.txt”

### 19. git diff

To figure out what changes you made since the last commit.

git diff

### 20. git reset head license

To undo the staging of the file that was added in the staging area.

git reset head license

### 21. git checkout –license

To Blow away all changes since the last commit of the file.

git checkout –license

### 22. git commit -a -m “ ”

To add any of our tracked files to the staging area and commit them by providing a message to remember.

git commit -a -m “Readme.md”

### 23. git reset –soft HEAD^

To undo the last commit and bring the file to the staging area.

git reset –soft HEAD^

### 24. git reset –hard HEAD^

To undo the last commit and remove the file from the staging area as well(In case we went horribly wrong).

git reset –hard HEAD^

### 25. git reset –hard HEAD^^

To undo the last 2 commits and all changes.

git reset –hard HEAD^^

### 26. git remote add origin

These commands make a bookmark which signifies that this particular remote refers to this URL. This remote will be used to pull any content from the directory and push our local content to the global server.

git remote add origin https://github.com/madaan123/MyAlgorithms.git

### 27. git remote add <address>

To add new remotes to our local repository for a particular git address.

git remote add <address> 

### 28. git remove rm

To remove a remote from our local repository.

git remove rm

### 29. git push -u origin master

To push all the contents of our local repository that belong to the master branch to the server(Global repository).

git push -u origin master

### 30. git clone https://github.com/madaan123/MyAlgorithms.git

To clone or make a local copy of the global repository in your system   
(git clone command downloads the repository and creates a remote named origin which can be checked by the command – git remote -v).

git clone https://github.com/madaan123/MyAlgorithms.git

### 31. git branch Testing

To create a new branch named Testing.

git branch Testing

### 32. git branch

To see all the branches present and current branches that we are working on.

git branch

### 33. git checkout Testing

To switch to branch Testing from the master branch.

git checkout Testing

### 34. ls

To see directories and files in the current directory.

ls

### 35. ls -la

To see hidden directories and files within the current directory.

ls -la

### 36. git merge Testing

To merge the Testing branch with the master branch.

git merge Testing

### 37. git branch -d Testing

To delete the Testing branch.

git branch -d Testing

### 38. git checkout -b admin

To create a new branch admin and set it as the current branch.

git checkout -b admin

### 39. git branch -r

To look at all the remote branches.

git branch -r

### 40. git branch -D Testing

To forcefully delete a branch without making commits.

git branch -D Testing

### 41. git tag

To see the list of available tags.

git tag

### 42. git checkout v0.0.1

To set the current tag to v0.0.1.

git checkout v0.0.1

### 43. git tag -a v0.0.3 -m “version 0.0.3”

To create a new tag.

git tag -a v0.0.3 -m “version 0.0.3”

### 44. git push –tags

To push the tags to the remote repository.

git push –tags

### 45. git fetch

To fetch down any changes from the global repository to the current repository.

git fetch

### 46. git stash

To move staged files to the stash area which is present in the staging area.

git stash

### 47. git stash pop

To get back the files that are present in the stash area.

git stash pop

### 48. git stash clear

To clear the stash folder.

git stash clear

### 49. git rebase

Three tasks are performed by git rebase

1. Move all changes to master which are not in origin/master to a temporary area.
2. Run all origin master commits.
3. Run all commits in the temporary area on top of our master one at a time, so it avoids merge commits.

git rebase

### 50. git –version

used to show the current version of Git

git –version

**What is Maven ?**

[Maven](https://maven.apache.org/) is an open-source [build automation](https://www.browserstack.com/guide/build-automation) and project management tool widely used for Java applications.  As a build automation tool, it automates the source code compilation and dependency management, assembles binary codes into packages, and executes test scripts. Maven translates and packages your source code so that it becomes an executable application.

* Around the globe in 2023, over [7000 enterprises](https://www.slintel.com/tech/build-automation/apache-maven-market-share) are currently  using Apache Maven as a build-automated tool.
* Apart from the build, it can handle team collaboration, reporting, documentation, distribution publication, and release management. Thus it has also become an excellent project management tool.

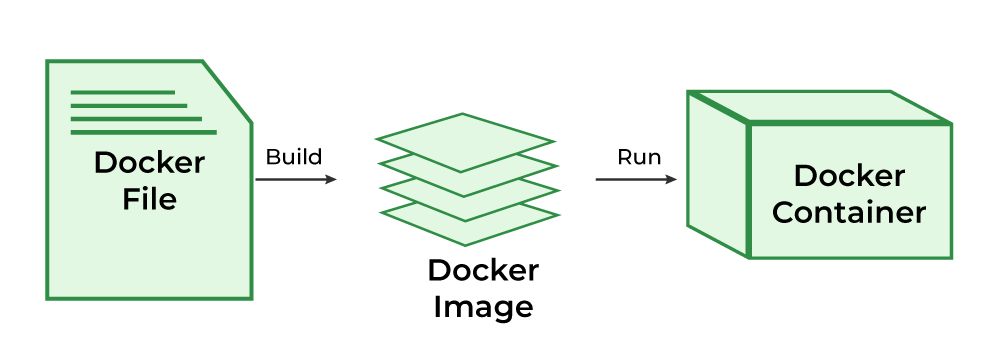
**What is use of Maven Product Management Tool ?**

Maven can be used for the following:

1. We can easily build a project using maven.
2. We can add jars and other dependencies of the project easily using the help of maven.
3. Maven provides project information (log document, dependency list, unit test reports, etc.)
4. Maven is very helpful for a project while updating the central repository of JARs and other dependencies.
5. With the help of Maven, we can build any number of projects into output types like the JAR, WAR, etc without doing any scripting.
6. Using maven we can easily integrate our project with a source control systems (such as Subversion or Git).
7. Maven also helps in managing the project’s build lifecycle, including tasks like compiling, testing, packaging, and deploying the code.
8. Maven provides a standard project structure, making it easy for developers to understand the layout of the project and locate specific files.
9. Maven supports multi-module projects, allowing developers to work on multiple related projects simultaneously and manage their dependencies efficiently.
10. Maven plugins can be used to add additional functionality to the build process, such as code coverage analysis, static code analysis, and more.
11. Maven is highly customizable, allowing developers to configure the build process to meet their specific needs and requirements.
12. Maven simplifies the process of managing project dependencies, ensuring that the correct versions of libraries and frameworks are used throughout the project.

**What is Docker File ?**

The operating system (OS) libraries and dependencies required to run the application source code which is not reliant on the underlying operating system (OS) included in the Dockerfile, which is a standardized, executable component. Programmers may design, distribute, launch, run, upgrade, and manage containers using the open-source platform Docker. Enterprise Edition (EE) and Community Edition (CE) of Docker are both available. The Enterprise Version is for businesses and IT teams working on mission-critical production applications, while the Community Edition is suitable for small teams just learning Docker.



## ****What is Docker Image?****

An artifact with several layers and a lightweight, compact stand-alone executable package that contains all of the components required to run a piece of software, including the code, a runtime, libraries, environment variables, and configuration files is called a [Docker image](https://www.geeksforgeeks.org/what-is-docker-images/).

**How to create Docker File ?**

1. **Start with a base image:** The first instruction in the Dockerfile should be the ‘FROM’ command, which specifies the base image to use. This can be an official image from the Docker Hub, such as ‘FROM ubuntu’, or a custom image built by another developer.
2. **Run commands:** Use the ‘RUN’ command to execute commands in the container. These commands are used to install dependencies, configure the environment, and perform other setup tasks.
3. **Copy files:** Use the ‘COPY’ command to copy files from the host machine into the container. This can be used to copy the application code or other configuration files.
4. **Set environment variables:** Use the ‘ENV’ command to set environment variables that will be available in the container.
5. **Expose ports:** Use the ‘EXPOSE’ command to indicate which ports the container should listen on.
6. **Set the default command:** Use the ‘CMD’ command to specify the command that should be run when the container starts.
7. **Build the image:** To build the image, navigate to the directory where the Dockerfile is located and run the following command: ‘docker build -t <image-name>’. This will create an image with the specified name, and the dot at the end specifies the current directory as the build context.

