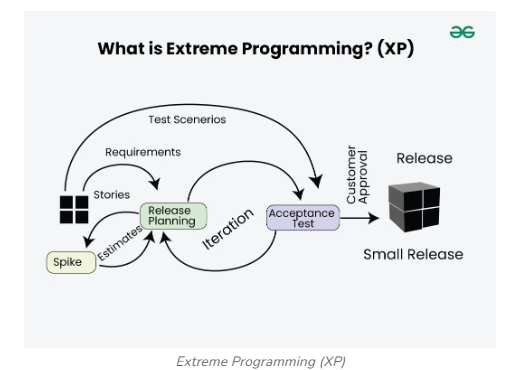
***Extreme Programming (XP)***

Extreme Programming (XP) is an [Agile software development](https://www.geeksforgeeks.org/software-engineering-agile-software-development/) methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation.

Extreme programming is one of the most popular and well-known approaches in the family of agile methods.

XP project starts with user stories which are short descriptions of what scenarios the customers and users would like the system to support. Each story is written on a separate card, so they can be flexibly grouped.

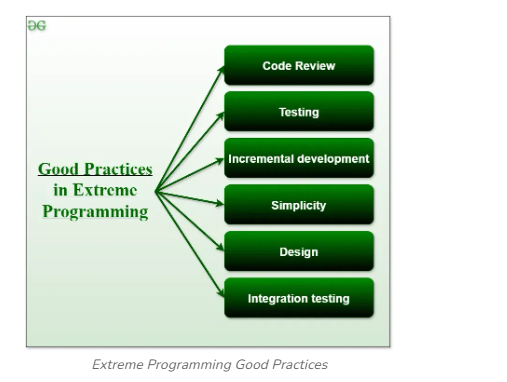


**Good Practices in Extreme Programming**

Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

*Extreme Programming Good Practices*

* **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their work between them every hour.
* **Testing:** [Testing](https://www.geeksforgeeks.org/software-testing-basics/)code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach, test cases are written even before any code is written.
* **Incremental development:**Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.
* **Simplicity:**Simplicity makes it easier to develop good-quality code as well as to test and debug it.
* **Design:**Good quality design is important to develop good quality software. So, everybody should design daily.
* **Integration testing:**[Integration Testing](https://www.geeksforgeeks.org/software-engineering-integration-testing/) helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.



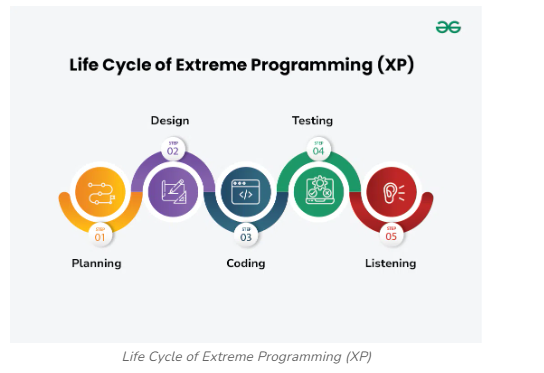
**Basic Principles of Extreme programming**

XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User Story is a conventional description by the user of a feature of the required system. It does not mention finer details such as the different scenarios that can occur. Based on User stories, the project team proposes Metaphors. Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some features. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype. Some of the basic activities that are followed during software development by using the XP model are given below:

* **Coding:** The concept of coding which is used in the XP model is slightly different from traditional coding. Here, the coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.
* **Testing:** The XP model gives high importance to testing and considers it to be the primary factor in developing fault-free software.
* **Listening:** The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, the programmers should understand properly the functionality of the system and they have to listen to the customers.
* **Designing:** Without a proper design, a system implementation becomes too complex, and very difficult to understand the solution, thus making maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.
* **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
* **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in the present time, rather than trying to build something that would take time and may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.
* **Pair Programming:**XP encourages [pair programming](https://www.geeksforgeeks.org/pair-programming/) where two developers work together at the same workstation. This approach helps in knowledge sharing, reduces errors, and improves code quality.
* **Continuous Integration:**In XP, developers integrate their code into a shared repository several times a day. This helps to detect and resolve integration issues early on in the development process.
* **Refactoring:**XP encourages [refactoring](https://www.geeksforgeeks.org/refactoring-introduction-and-its-techniques/), which is the process of restructuring existing code to make it more efficient and maintainable. Refactoring helps to keep the codebase clean, organized, and easy to understand.
* **Collective Code Ownership:** In XP, there is no individual ownership of code. Instead, the entire team is responsible for the codebase. This approach ensures that all team members have a sense of ownership and responsibility towards the code.
* **Planning Game:** XP follows a planning game, where the customer and the development team collaborate to prioritize and plan development tasks. This approach helps to ensure that the team is working on the most important features and delivers value to the customer.
* **On-site Customer:** XP requires an on-site customer who works closely with the development team throughout the project. This approach helps to ensure that the customer’s needs are understood and met, and also facilitates communication and feedback.

**Life Cycle of Extreme Programming (XP)**

The Extreme Programming Life Cycle consist of five phases:



*Life Cycle of Extreme Programming (XP)*

1. **Planning:**The first stage of Extreme Programming is planning. During this phase, clients define their needs in concise descriptions known as user stories. The team calculates the effort required for each story and schedules releases according to priority and effort.
2. **Design:**The team creates only the essential design needed for current user stories, using a common analogy or story to help everyone understand the overall system architecture and keep the design straightforward and clear.
3. **Coding:**Extreme Programming (XP) promotes pair programming i.e**.**wo developers work together at one workstation, enhancing code quality and knowledge sharing.
4. **Testing:**Extreme Programming (XP) gives more importance to testing that consist of both unit tests and acceptance test**.**Unit tests, which are automated, check if specific features work correctly. Acceptance tests, conducted by customers, ensure that the overall system meets initial requirements. This continuous testing ensures the software’s quality and alignment with customer needs.
5. **Listening:** In the listening phase regular feedback from customers to ensure the product meets their needs and to adapt to any changes.

**Values of Extreme Programming (XP)**

There are five core values of Extreme Programming (XP)



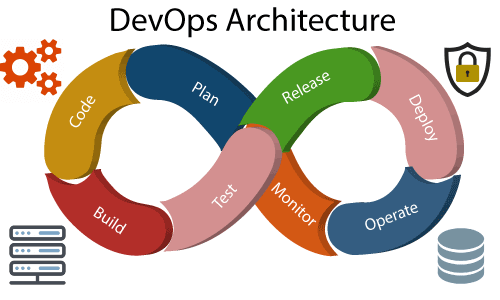
*Values of Extreme Programming (XP)*

1. **Communication:**The essence of communication is for information and ideas to be exchanged amongst development team members so that everyone has an understanding of the system requirements and goals. Extreme Programming (XP) supports this by allowing open and frequent communication between members of a team.
2. **Simplicity:**Keeping things as simple as possible helps reduce complexity and makes it easier to understand and maintain the code.
3. **Feedback:**Feedback loops which are constant are among testing as well as customer involvements which helps in detecting problems earlier during development.
4. **Courage:**Team members are encouraged to take risks, speak up about problems, and adapt to change without fear of repercussions.
5. **Respect**: Every member’s input or opinion is appreciated which promotes a collective way of working among people who are supportive within a certain group.

**Advantages of Extreme Programming (XP)**

* **Slipped schedules:**Timely delivery is ensured through slipping timetables and doable development cycles.
* **Misunderstanding the business and/or domain −** Constant contact and explanations are ensured by including the client on the team.
* **Canceled projects:**Focusing on ongoing customer engagement guarantees open communication with the consumer and prompt problem-solving.
* **Staff turnover:**Teamwork that is focused on cooperation provides excitement and goodwill. Team spirit is fostered by multidisciplinary cohesion.
* **Costs incurred in changes:** Extensive and continuing testing ensures that the modifications do not impair the functioning of the system. A functioning system always guarantees that there is enough time to accommodate changes without impairing ongoing operations.
* **Business changes:** Changes are accepted at any moment since they are seen to be inevitable.
* **Production and post-delivery defects:** the unit tests to find and repair bugs as soon as possible.

# **DevOps Architecture**

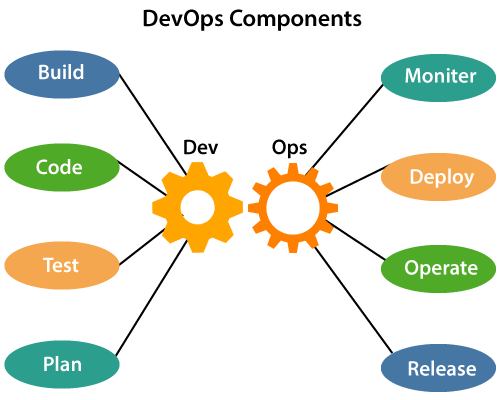


Development and operations both play essential roles in order to deliver applications. The deployment comprises analyzing the **requirements, designing, developing**, and **testing** of the software components or frameworks.

The operation consists of the administrative processes, services, and support for the software. When both the development and operations are combined with collaborating, then the DevOps architecture is the solution to fix the gap between deployment and operation terms; therefore, delivery can be faster.

DevOps architecture is used for the applications hosted on the cloud platform and large distributed applications. Agile Development is used in the DevOps architecture so that integration and delivery can be contiguous. When the development and operations team works separately from each other, then it is time-consuming to **design, test**, and **deploy**. And if the terms are not in sync with each other, then it may cause a delay in the delivery. So DevOps enables the teams to change their shortcomings and increases productivity.

Below are the various components that are used in the DevOps architecture:



### 1) Build

Without DevOps, the cost of the consumption of the resources was evaluated based on the pre-defined individual usage with fixed hardware allocation. And with DevOps, the usage of cloud, sharing of resources comes into the picture, and the build is dependent upon the user's need, which is a mechanism to control the usage of resources or capacity.

### 2) Code

Many good practices such as Git enables the code to be used, which ensures writing the code for business, helps to track changes, getting notified about the reason behind the difference in the actual and the expected output, and if necessary reverting to the original code developed. The code can be appropriately arranged in **files, folders**, etc. And they can be reused.

### 3) Test

The application will be ready for production after testing. In the case of manual testing, it consumes more time in testing and moving the code to the output. The testing can be automated, which decreases the time for testing so that the time to deploy the code to production can be reduced as automating the running of the scripts will remove many manual steps.

### 4) Plan

DevOps use Agile methodology to plan the development. With the operations and development team in sync, it helps in organizing the work to plan accordingly to increase productivity.

### 5) Monitor

Continuous monitoring is used to identify any risk of failure. Also, it helps in tracking the system accurately so that the health of the application can be checked. The monitoring becomes more comfortable with services where the log data may get monitored through many third-party tools such as **Splunk**.

### 6) Deploy

Many systems can support the scheduler for automated deployment. The cloud management platform enables users to capture accurate insights and view the optimization scenario, analytics on trends by the deployment of dashboards.

ADVERTISEMENT

ADVERTISEMENT

### 7) Operate

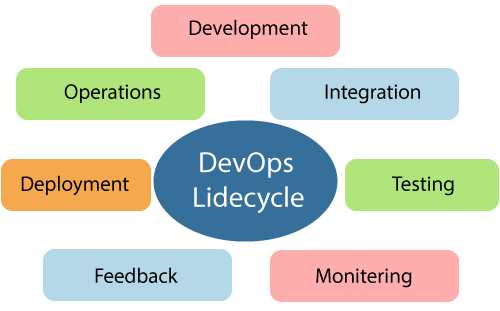
DevOps changes the way traditional approach of developing and testing separately. The teams operate in a collaborative way where both the teams actively participate throughout the service lifecycle. The operation team interacts with developers, and they come up with a monitoring plan which serves the IT and business requirements.

### 8) Release

Deployment to an environment can be done by automation. But when the deployment is made to the production environment, it is done by manual triggering. Many processes involved in release management commonly used to do the deployment in the production environment manually to lessen the impact on the customers.

**DevOps Lifecycle**

DevOps defines an agile relationship between operations and Development. It is a process that is practiced by the development team and operational engineers together from beginning to the final stage of the product.



Learning DevOps is not complete without understanding the DevOps lifecycle phases. The DevOps lifecycle includes seven phases as given below:

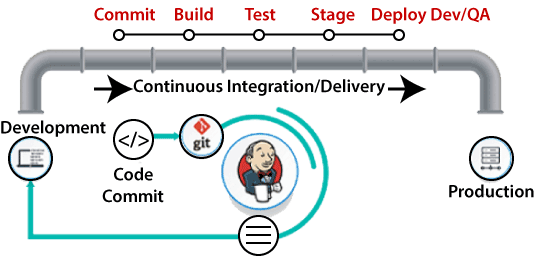
**1) Continuous Development**

This phase involves the planning and coding of the software. The vision of the project is decided during the planning phase. And the developers begin developing the code for the application. There are no DevOps tools that are required for planning, but there are several tools for maintaining the code.

**2) Continuous Integration**

This stage is the heart of the entire DevOps lifecycle. It is a software development practice in which the developers require to commit changes to the source code more frequently. This may be on a daily or weekly basis. Then every commit is built, and this allows early detection of problems if they are present. Building code is not only involved compilation, but it also includes **unit testing, integration testing, code review**, and **packaging**.

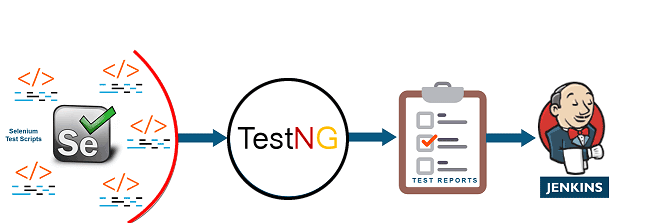
The code supporting new functionality is continuously integrated with the existing code. Therefore, there is continuous development of software. The updated code needs to be integrated continuously and smoothly with the systems to reflect changes to the end-users.



Jenkins is a popular tool used in this phase. Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a build of that code, which is an executable file in the form of war or jar. Then this build is forwarded to the test server or the production server.

**3) Continuous Testing**

This phase, where the developed software is continuously testing for bugs. For constant testing, automation testing tools such as **TestNG, JUnit, Selenium**, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality. In this phase, **Docker** Containers can be used for simulating the test environment.



**Selenium** does the automation testing, and TestNG generates the reports. This entire testing phase can automate with the help of a Continuous Integration tool called **Jenkins**.

Automation testing saves a lot of time and effort for executing the tests instead of doing this manually. Apart from that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. Also, we can schedule the execution of the test cases at predefined times. After testing, the code is continuously integrated with the existing code.

**4) Continuous Monitoring**

Monitoring is a phase that involves all the operational factors of the entire DevOps process, where important information about the use of the software is recorded and carefully processed to find out trends and identify problem areas. Usually, the monitoring is integrated within the operational capabilities of the software application.

It may occur in the form of documentation files or maybe produce large-scale data about the application parameters when it is in a continuous use position. The system errors such as server not reachable, low memory, etc are resolved in this phase. It maintains the security and availability of the service.

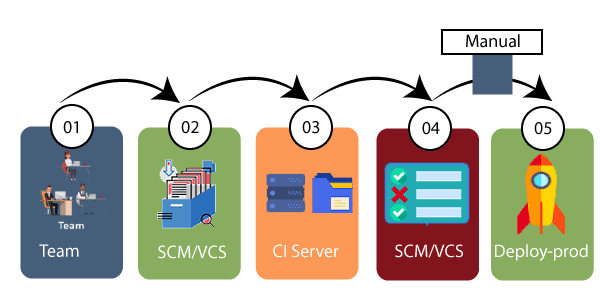
**5) Continuous Feedback**

The application development is consistently improved by analyzing the results from the operations of the software. This is carried out by placing the critical phase of constant feedback between the operations and the development of the next version of the current software application.

The continuity is the essential factor in the DevOps as it removes the unnecessary steps which are required to take a software application from development, using it to find out its issues and then producing a better version. It kills the efficiency that may be possible with the app and reduce the number of interested customers.

**6) Continuous Deployment**

In this phase, the code is deployed to the production servers. Also, it is essential to ensure that the code is correctly used on all the servers.



The new code is deployed continuously, and configuration management tools play an essential role in executing tasks frequently and quickly. Here are some popular tools which are used in this phase, such as **Chef, Puppet, Ansible**, and **SaltStack**.

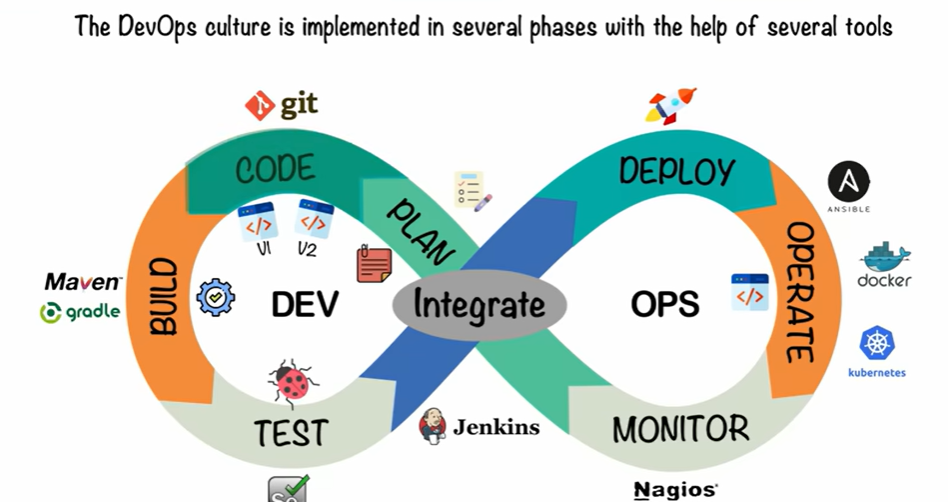
Containerization tools are also playing an essential role in the deployment phase. **Vagrant** and **Docker** are popular tools that are used for this purpose. These tools help to produce consistency across development, staging, testing, and production environment. They also help in scaling up and scaling down instances softly.

Containerization tools help to maintain consistency across the environments where the application is tested, developed, and deployed. There is no chance of errors or failure in the production environment as they package and replicate the same dependencies and packages used in the testing, development, and staging environment. It makes the application easy to run on different computers.

**7) Continuous Operations**

All DevOps operations are based on the continuity with complete automation of the release process and allow the organization to accelerate the overall time to market continuingly.

It is clear from the discussion that continuity is the critical factor in the DevOps in removing steps that often distract the development, take it longer to detect issues and produce a better version of the product after several months. With DevOps, we can make any software product more efficient and increase the overall count of interested customers in your product.



**What is Git?**

Git is a popular version control system.

It is used for:

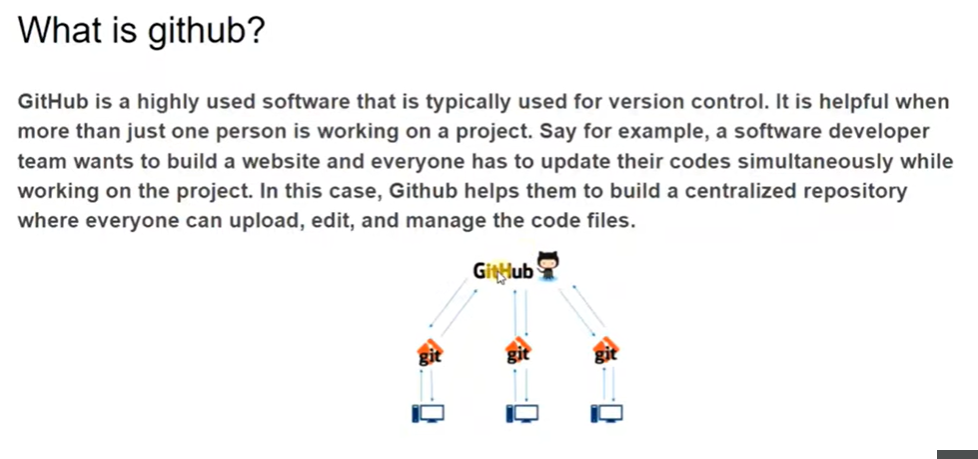
* Tracking code changes
* Tracking who made changes
* Coding collaboration

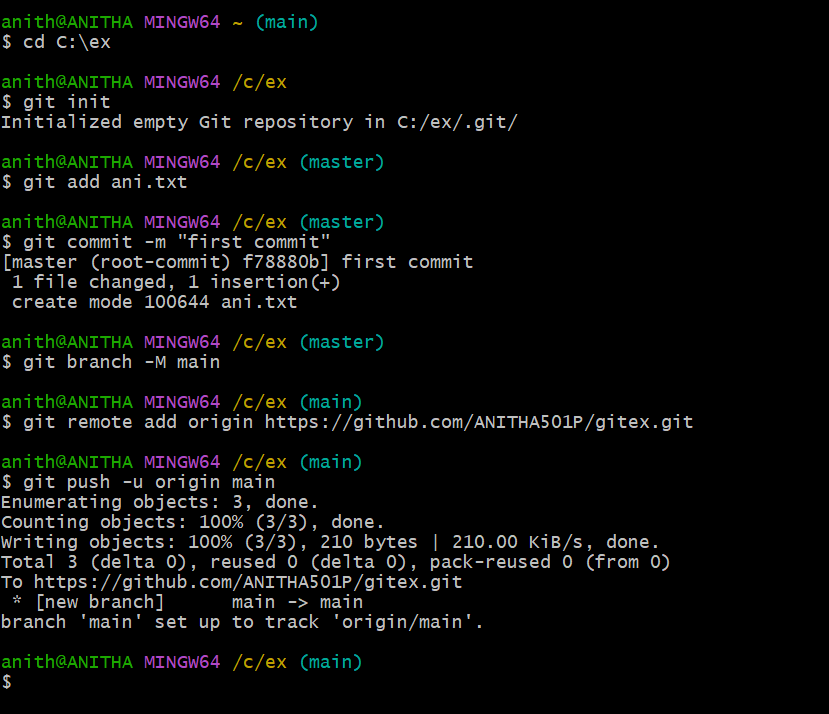
What does Git do?

* Manage projects with **Repositories**
* **Clone** a project to work on a local copy
* Control and track changes with **Staging** and **Committing**
* **Branch** and **Merge** to allow for work on different parts and versions of a project
* **Pull** the latest version of the project to a local copy
* **Push** local updates to the main project

[**GitHub**](https://github.com/)

[**GitHub**](https://github.com/) is a cloud software development platform. It is commonly used for saving files, tracking changes, and collaborating on development projects.





<https://www.youtube.com/watch?v=XdMMfUKBbtE&list=PLzdWZT-ZJD081YB1TLN5rNI1vw5hTdRzH>

(ex-5)<https://www.youtube.com/watch?v=31igoWxauEQ>