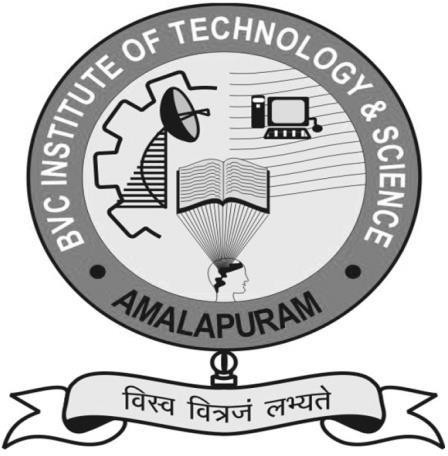
###### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

###### (ARTIFICIAL INTELLIGENCE AND DATA SCIENCE)

**CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY USING DevOps LAB MANUAL**

**STUDENT LAB MANUAL**

**III/I CAD (R20)**

###### B V C INSTITUTE OF TECHNOLOGY AND SCIENCE AMALAPURAM

**(Permanently Affiliated to JNTUK Kakinada, Accredited by NAAC, Approved by AICTE New Delhi)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INDEX** | | | | |
|  | **S.NO** | **DESCRIPTION** | **Page**  **No.** |  |
| 1 | Institute Vision and Mission | 5 |
| 2 | Department Vision and Mission | 6 |
| 3 | PO’s | 7 |
| 4 | PSO’s, PEO’s | 8 |
| 5 | Course Outcomes | 9 |
| 6 | CO/ PO/ PSO Justification | 10 |
| 7 | University / JNTUK Syllabus | 11 |
|  | **LIST OF EXPERIMENTS** |  |
| 1 | **Software engineering and Agile software development:-**Get an understanding of the stages in software development lifecycle, the process models, values and principles of agility and the need for agile software development. This will enable you to work in projects following an agile approach to software development. |  |
| 2 | **Development & Testing with Agile: Extreme Programming:-**  Get a working knowledge of using extreme automation through XP programming practices of test first development, refactoring and automating test case writing. |  |
| 3 | **DevOps adoption in projects :-** It is important to comprehend the need to automate the software development lifecycle stages through DevOps. Gain an understanding of the capabilities required to implement DevOps, continuous integration and continuous delivery practices. |  |
| 4 | **Implementation of CICD with Java and open source stack:-** Configure the web application and Version control using Git using Git commands and version control operations. |  |
| 5 | Configure a static code analyzer which will perform static analysis of the web application code and identify the coding practices that are not appropriate. Configure the profiles and dashboard of the static code analysis tool. |  |
| 6 | Write a build script to build the application using a build automation tool like Maven. Create a folder structure that will run the build script and invoke the various software development build stages. This script should invoke the static analysis tool and unit test cases and deploy the application to a web application server like Tomcat. |  |
| 7 | Configure the Jenkins tool with the required paths, path variables, users and pipeline views |  |
| 8 | Configure the Jenkins pipeline to call the build script jobs and configure to run it whenever there is a change made to an application in the version control system. Make a change to the background color of the landing page of the web application and check if the configured pipeline runs. |  |
| 9 |  |  |
| 10 | Create a pipeline view of the Jenkins pipeline used in Exercise 8. Configure it with user defined messages. |  |
| 11 | In the configured Jenkins pipeline created in Exercise 8 and 9, implement quality gates for static analysis of code. |  |
| 12 | In the configured Jenkins pipeline created in Exercise 8 and 9, implement quality gates for static unit testing |  |
| 13 | In the configured Jenkins pipeline created in Exercise 8 and 9, implement quality gates for code coverage. |  |
|  |  |  |
|  |  |  |

**INSTITUTE VISION AND MISSION**

**Vision**

To be a premier institution in Education and Research, producing global leaders in Engineering Technology and management.

**Mission**

**IM1:** Imparting quality and outcome based education towards academic excellence.

**IM2:** Inculcate team spirit and professional ethics among stakeholders.

**IM3:** Strengthen links with industry through internships and collaborative development works.

###### DEPARTMENT VISION AND MISSION

**Vision**

To become a prominent department of Computer Science and Engineering producing competent professionals with research and innovation skills, inculcating moral values and societal concerns.

###### Mission

**DM 1:** To offer state –of-art education in Computer Science and Engineering.

**DM2:** To provide strong theoretical foundation complemented with extensive practical training.

**DM 3:** To train and transform young men and women into responsible thinking engineers, technologists and scientists, to motivate them to attain professional excellence.

**DM 4:** To inspire students proactively engage themselves for the betterment of the society

Head of the Department

###### PROGRAM OUTCOMES (POs)

**PO1 Engineering Knowledge:** Apply knowledge of mathematics and science, with fundamentals of Computer Science and Engineering to be able to solve complex engineering problems related to CSE.

**PO2 Problem Analysis:** Identify, Formulate, review research literature and analyze complex engineering problems related to Computer Science and Engineering and reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences

**PO3 Design/Development of solutions:** Design solutions for complex engineering problems related to Computer Science and Engineering and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural societal and environmental considerations

**PO4 Conduct Investigations of Complex problems:** Use research–based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5 Modern Tool Usage:** Create, Select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to computer science related complex engineering activities with an understanding of the limitations

**PO6 The Engineer and Society:** Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the Computer Science and Engineering professional engineeringpractice

**PO7 Environment and Sustainability:** Understand the impact of the Computer Science and Engineering professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development

**PO8 Ethics:** Apply Ethical Principles and commit to professional ethics and responsibilities and norms of the engineering practice

**PO9 Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary Settings

6

**PO10 Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large such as able to comprehend and with write effective reports and design documentation, make effective presentations and give and receive clear instructions.

**PO11 Project Management and Finance:** Demonstrate knowledge and understanding of the engineering management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments

**PO12 Life-Long Learning:** Recognize the need for and have the preparation and abilityto engage in independent and life-long learning the broadest context of technological change.

are:

###### PROGRAM EDUCATIONAL OBJECTIVES (PEO’s)

The educational objectives of UG program in Computer Science and Engineering

**PEO - 1:** Graduates will be an efficient software developer in diverse fields and will be a successful professional; and /or persue higher studies

**PEO - 2:** Graduates will be capable to adapt to new computing technology for professional excellence and research and be a lifelong learner.

**PEO 3:** Graduates will work productively exhibiting ethical qualities for the betterment of society.

**PEO 4:** Graduates will possess leadership qualities, work harmoniously as a team member with effective communication skills.

###### PROGRAM SPECIFIC OUTCOMES (PSO’s)

The Computer Science and Engineering Program will demonstrate:

**PSO1:** Gain capability to use current techniques, skills and tools necessary for carrying out multi disciplinary projects

**PSO2:** acquaint with the contemporary trending industrial / research setting and their by innovate novel solutions to existing problems.

**1. Get an understanding of the stages in software development lifecycle, the process models, values and principles of agility and the need for agile software development. This will enable you to work in projects following an agile approach to software development.**

**Software engineering and Agile software development**

**Traditional Software Development:** Traditional software development is the software development process used to design and develop simple software. It is used when the security and many other factors of the software are not much important. It is used by fresher to develop the software. It consists of five phases:

**1.** Requirements analysis

1.Requirement Analysis

2.Design

3.Implementation

4.Coding And Testing

5.Maintenance

**Agile software development**

Agile software development is the software development process used to design complicated software. It is used when the software is quite sensitive and complicated. It is used when security is much more important. It is used by professionals to develop the software. It consists of three phases

1.Project Initiation

2.Sprint Planning

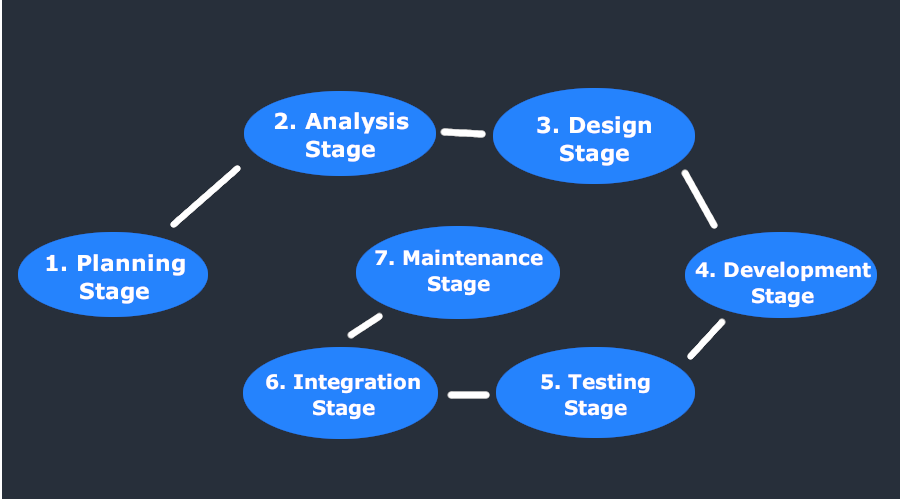
3.Demos

**Difference** **between** **Traditional and Agile Software Development**:

| S. No. | Traditional Software Development | Agile Software Development |
| --- | --- | --- |
| 1. | It is used to develop simple software. | It is used to develop complicated software. |
| 2. | In this methodology, testing is done once the development phase is completed. | In this methodology, testing and development processes are performed concurrently. |
| 3. | It follows a linear organization structure. | It follows an iterative organizational structure. |
| 4. | It provides less security. | It provides high security. |
| 5. | Client involvement is less as compared to Agile development. | Client involvement is high as compared to traditional software development. |
| 6. | It provides less functionality in the software. | It provides all the functionality needed by the users. |
| 7. | It supports a fixed development model. | It supports a changeable development model. |
| 8. | It is used by freshers. | It is used by professionals. |
| 9. | Development cost is less using this methodology. | Development cost is high using this methodology. |
| 10. | It majorly consists of five phases. | It consists of only three phases. |
| 11. | It is less used by software development firms. | It is normally used by software development firms. |
| 12. | Expectation is favored in the traditional model. | Adaptability is favored in the agile methodology. |

|  |  |  |
| --- | --- | --- |
| 13. | Traditional software development approaches are formal in terms of communication with customers. | Agile software development methodologies are casual. In other words, customers who work with companies that utilize Agile software development approaches are more likely to interact with them than customers who work with companies that use traditional software development methodology. |
| 14. | For starters, typical software development approaches employ a predictive approach. There is full specification and prediction of the software development processes because the product is produced through rigorous and explicit planning. Changes are not permitted in this technique because the time and cost of project development are fixed. | Here, a flexible approach is used as the software development approaches are founded on the notion of continual design improvement and testing relies on team and client feedback. |
| 15. | **Examples-**   * Office productivity suites * Data management software * Media players * Security programs | **Examples-**   * Sky * Phillips * JP Morgan Chase |
| 16. | **Models based on Traditional Software Development-**   * Spiral Model * Waterfall Model * V Model * Incremental Model | **Models based on Agile Software Development-**   * Scrum * Extreme Programming (XP) * Crystal * Dynamic Systems Development Method (DSDM) * Feature Driven Development (FDD) * Adaptive Software Development (ASD) |

Software development life cycle :



**What is System Development Life Cycle?**

A system development life cycle or SDLC is essentially a project management model. It defines different stages that are necessary to bring a project from its initial idea or conception all the way to deployment and later maintenance.

**7 Stages of the System Development Life Cycle**

There are seven primary stages of the modern system development life cycle. Here’s a brief breakdown:

* Planning Stage
* Feasibility or Requirements of Analysis Stage
* Design and Prototyping Stage
* Software Development Stage
* Software Testing Stage
* Implementation and Integration
* Operations and Maintenance Stage

### Phase 1: Planning

The planning phase will determine project goals and establish a high-level plan for the intended project. Planning is, by definition, a fundamental and critical organizational phase. The three primary activities involved in the planning phase are as follows:

1. Identification of the system for development
2. Feasibility assessment
3. Creation of project plan

### Phase 2: Analysis

End-user business requirement analysis takes place during this phase. Project goals are converted into the defined system functions that the organization intends to develop. The three primary activities involved in the analysis phase are as follows:

1. Gathering business requirement
2. Creating process diagrams
3. Performing a detailed analysis

Business requirement gathering is the most crucial part at this level of SDLC. Business requirements are a brief set of business functionalities that the system needs to meet to be successful. This phase does not define technical details such as the type of technology implemented in the system. A sample business requirement might look like “The system must track all the employees by their respective department, region, and the designation.” This requirement shows no such detail as to how the system will implement this requirement, but rather what the system must do concerning the business.

### Phase 3: Design

In the design phase, we describe the desired features and operations of the system. This phase includes business rules, pseudo-code, screen layouts, and other necessary documentation. The two primary activities involved in the design phase are as follows:

1. Designing the IT infrastructure
2. Designing the system model

The IT infrastructure should have solid foundations to avoid any crash, malfunction, or reduction in performance.  In this phase, the specialist recommends the clients and servers required on a cost and time basis and the system’s technical feasibility. The organization also creates user interaction interfaces, data models, and entity relationship diagrams (ERDs) in this phase.

### Phase 4: Development

The development phase is when all documents from the previous step transfer into the system. The primary activities involved in the development phase are as follows:

1. Development of IT infrastructure
2. Development of database and code

During the design phase, just the IT infrastructure blueprint is provided. In contrast, the organization purchases and installs the respective software and hardware to support the IT infrastructure during the development phase. Following this, creating the database and actual code can begin to complete the system according to the specifications.

### Phase 5: Testing

All pieces of code are integrated during the testing phase and deployed in the testing environment. Testers then work through [Software Testing Life Cycle](https://blog.testlodge.com/software-testing-life-cycle/) activities to check the system for errors, bugs, and defects to verify the system’s functionalities work as expected. The two primary activities involved in the testing phase are as follows:

1. Writing test cases
2. Execution of test cases

Testing is a critical part of the software development life cycle. To provide quality software, an organization must systematically perform testing. After writing test cases, the tester executes them. They compare the expected result with an actual result to verify the system and ensure it operates correctly. Writing test cases and manually performing them is an intensive task for any organization but will succeed if executed properly.

### Phase 6: Deployment

During this next phase, the system is deployed to a real-life (the client’s) environment where actual users can begin operating the system. All data and components are present in the production environment. This phase is also called ‘delivery.’

### Phase 7: Maintenance

Any necessary enhancements, corrections, and changes are made during the maintenance phase to ensure the system continues to work and remain updated to meet business goals. It is necessary to maintain and upgrade the system from time to time to adapt to future needs. The three primary activities involved in the maintenance phase are as follows:

1. Support the system users
2. System maintenance
3. System changes and adjustment

**2.Development & Testing with Agile: Extreme Programming**

Get a working knowledge of using extreme automation through XP programming practices of test first  
development, refactoring and automating test case writing.

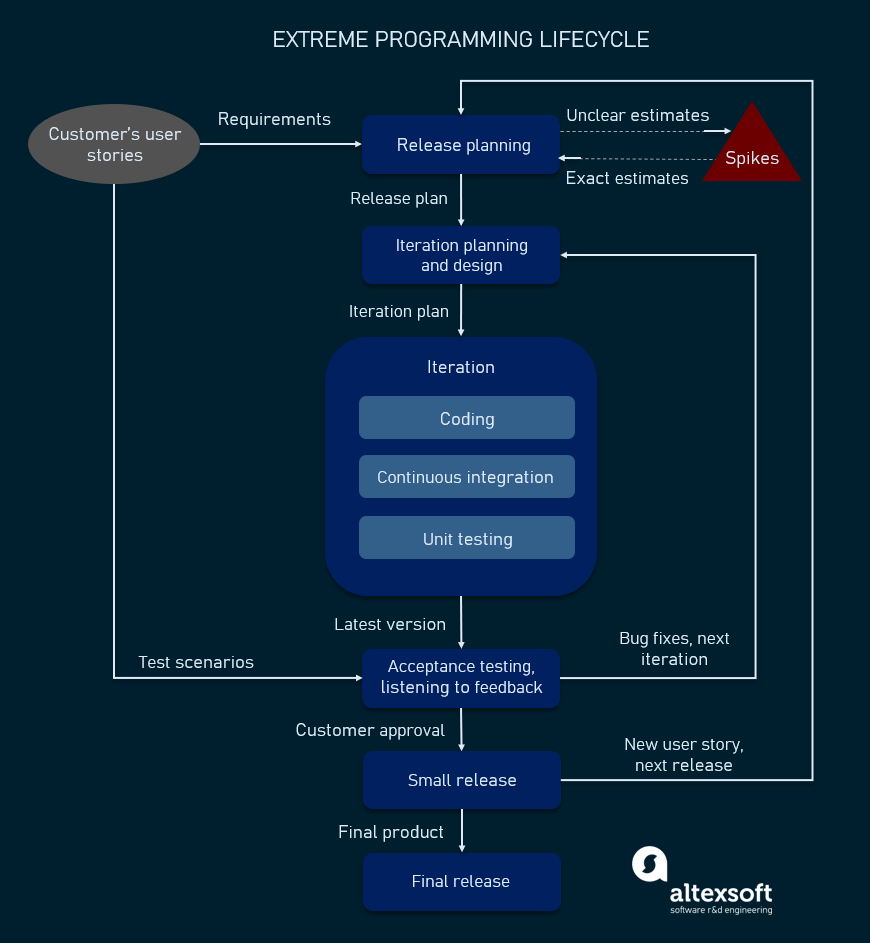
**Extreme Programming (XP)** is one of the numerous Agile frameworks applied by IT companies. But its key feature — emphasis on technical aspects of software development — distinguishes XP from the other approaches.

Software engineer [Ken Beck](https://en.wikipedia.org/wiki/Kent_Beck) introduced XP in the 90s with the goal of finding ways to write high-qualitative software quickly and being able to adapt to customers’ changing requirements.

**The process and roles of extreme programming:-**

The XP framework normally involves 5 phases or stages of the development process that iterate continuously:

1. **Planning,**the first stage, is when the customer meets the development team and presents the [requirements](https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/) in the form of user stories to describe the desired result.
2. **Designing**is actually a part of the planning process, but can be set apart to emphasize its importance. A good design brings logic and structure to the system and allows to avoid unnecessary complexities and redundancies.
3. **Coding**is the phase during which the actual code is created by implementing specific XP practices such as coding standards, pair programming, continuous integration, and collective code ownership
4. **Testing**is the core of extreme programming. It is the regular activity that involves both unit tests ([automated testing](https://www.altexsoft.com/whitepapers/quality-assurance-quality-control-and-testing-the-basics-of-software-quality-management/) to determine if the developed feature works properly) and acceptance tests (customer testing to verify that the overall system is created according to the initial requirements).
5. **Listening**is all about constant communication and feedback. The customers and project managers are involved to describe the business logic and value that is expected.



Such a development process entails the cooperation between several participants, each having his or her own tasks and responsibilities.

1. **Customers**are expected to be heavily engaged in the development process by creating user stories, providing continuous feedback, and making all the necessary business decisions related to the project.
2. **Programmers or developers**are the team members that actually create the product.
3. **Tester** responsible for implementing user stories and conducting user tests
4. **Trackers or managers** link customers and developers. These people organize the meetups, regulate discussions, and keep track of important progress KPIs.
5. **Coaches** can be included in the teams as mentors to help with understanding the XP practices.

## Values and principles of extreme programming:-

## XP values and principles

### Values of extreme programming

XP has simple rules that are based on 5 values to guide the teamwork:

1. **Communication.** Everyone on a team works jointly at every stage of the project.
2. **Simplicity.** Developers strive to write simple code bringing more value to a product, as it saves time and effort.
3. **Feedback.** Team members deliver software frequently, get feedback about it, and improve a product according to the new requirements.
4. **Respect.** Every person assigned to a project contributes to a common goal.
5. **Courage.** Programmers objectively evaluate their own results without making excuses and are always ready to respond to changes.

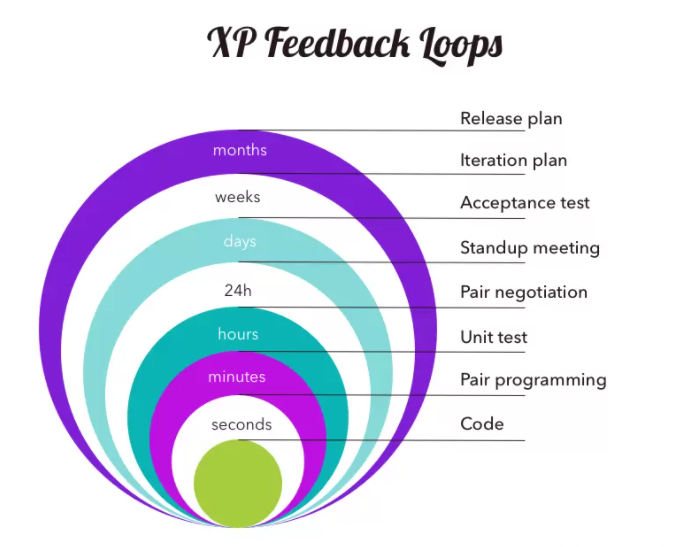
### Principles of extreme programming

Most researchers denote 5 XP principles as:

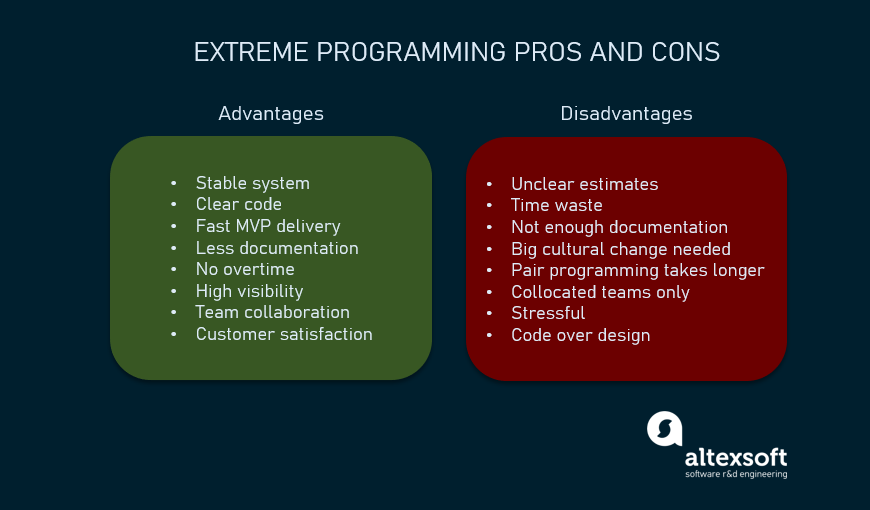
1. **Rapid feedback.** Team members understand the given feedback and react to it right away.
2. **Assumed simplicity.** Developers need to focus on the job that is important at the moment and follow YAGNI (You Ain’t Gonna Need It) and DRY (Don’t Repeat Yourself) principles.
3. **Incremental changes.** Small changes made to a product step by step work better than big ones made at once.
4. **Embracing change.** If a client thinks a product needs to be changed, programmers should support this decision and plan how to implement new requirements.
5. **Quality work.** A team that works well, makes a valuable product and feels proud of it.

## Extreme Programming Practices

Pair programming in XP iteration cycle, source: extremeprogramming.org



## Advantages and disadvantages of XP



## Simple Design – Support from other XP Practices

**Simple Design − Disadvantages**

You cannot possibly have just enough design for today's code and your design may not continue to evolve the system.

### Simple Design with other XP practices.

The other XP practices support Simple Design in the following way −

* Refactoring allows you to make changes.
* Pair programming helps you to be confident that you are making a simple design that works.
* 40-hour week helps you to be focused on the right design.
* Continuous unit testing and customer testing ensures that your simple design is on the track.

## Refactoring – Support from other XP Practices

### Refactoring - Disadvantages

You cannot possibly **refactor** the design of the system all the time. It would −

* Take too long.
* Be too hard to control, and
* Most likely break the system.

### Refactoring with other XP practices

The other XP practices support Refactoring in the following way −

* With collective ownership, you can make changes wherever they are needed.
* With coding standards, you need not reformat before refactoring.
* With pair programming, you can have the courage to tackle a tough refactoring.
* With a simple design, the refactoring is easier.
* With metaphor, you can communicate easily.
* With testing, you are less likely to break something without knowing it.
* With continuous integration, if you accidentally break something, or if your refactoring conflicts with someone else's work, you come to know in a few hours.
* With 40-hour week, you are rested and so you have more courage and are less likely to make mistakes.

Thus, you can **refactor** whenever you see the chance to −

* Make the system simpler
* Reduce duplication
* Communicate more clearly

**3.DevOps adoption in projects :- It is important to comprehend the need to automate the software development lifecycle stages through DevOps. Gain an understanding of the capabilities required to implement DevOps, continuous integration and continuous delivery practices**

DevOps combines the words "development" and "operations," and it refers to a collaborative approach to building applications. The goal of DevOps is to produce better, more reliable software products.

## How does DevOps work?

DevOps influences the application lifecycle. It typically has four phases, which work in a loop.Each phase depends on the other phases, and they aren't role-specific. That means that each person on the team is involved in multiple phases of the lifecycle (or possibly all of them). The four phases are:

* **Plan:**This is when DevOps teams define the features and capabilities of the applications and systems they're working on. They decide how to track progress and bugs and break the work into small, manageable pieces.
* **Develop:**This is when the coding happens, including writing, testing, reviewing, and integration. The goal is to work rapidly without sacrificing quality and stability. To facilitate development, DevOps teams use productivity tools and automation.
* **Deliver:**Delivery is when the application is moved into a production environment. The DevOps team determines a release schedule and uses automation to move applications from one stage to another.
* **Operate**. This involves maintaining, monitoring, and troubleshooting applications once they go live. Teams work to minimize downtime while also keeping an eye on security and compliance. The goal in the operating phase is to catch any issues before they impact customers and implement solutions. This leads back to the planning phase, and the cycle continues.

## DevOps practices

DevOps also incorporates several practices. These include:

* Continuous integration/continuous delivery (CI/CD)
* Microservices
* Monitoring and logging
* Communication and collaboration

Let's take a closer look at each of these practices.

### Continuous integration and continuous delivery (CI/CD)

Continuous integration is a software development practice where developers merge their code changes into a central repository, which ensures there aren't too many branches of an app in development.

Continuous delivery means that changes to an application are automatically tested for bugs and uploaded to a repository where they can be deployed by the operations team.

### Microservices

This Practice Structures An Application As A Collection Of Smaller Services. These Services Are Easy To Maintain, Testable, Independently Deployable, And Owned By A Small Team.

### Monitoring And Logging

This Involves Tracking Metrics And Keeping Logs To See How The Application Performs. This Data Is Analyzed To Find The Root Causes Of Problems And Unexpected Changes. Active Monitoring Allows Engineers To Be Proactive About Any Issues, Which Helps Minimize Downtime.

### Communication And Collaboration

For Devops To Be Successful, Everyone Involved Has To Be Committed To Sharing Information. This Might Be Done Through Chat, Project Tracking Systems, Or Wikis.

## What are the benefits of DevOps?

DevOps offers several benefits to organizations:

* **Speed:**DevOps increases the frequency of releases so you can improve your product quickly. The sooner a product is updated, the happier your customers will be.
* **Adaptability:**Technology and consumer expectations are constantly changing. DevOps allows you to pivot easily to incorporate different features and move in different directions without disrupting the entire development process.
* **Reliability:** Having more people involved in the entire process ensures a higher quality product. CI/CD is an important DevOps practice that helps teams roll out changes in a controlled way and avoid configuration drift, which is when configuration deviates over time. All of that results in a more stable, more reliable product.
* **Improved collaboration**. DevOps is a change to the culture of a company, emphasizing accountability and ownership. Teams are expected to collaborate and are given the tools to do so. DevOps teams share responsibilities and workflows, which saves time and reduces inefficiencies.

## What are the challenges of DevOps?

While DevOps can help organizations, it also has its challenges. They include:

* Implementing expensive tools and platforms
* Learning new tools
* Updating old infrastructure
* Integrating tools across different departments
* Reviewing current processes to make them more efficient
* Resistance to change
* Merging two cultures (development and operations) can be challenging
* Moving at a faster pace can make it difficult to maintain security and compliance

All of these issues can be overcome with planning, implementing DevOps over time, seeking input from team members, and dedicating time and resources to security and compliance throughout the DevOps lifecycle.

Here are some of the skills required to succeed in a DevOps career:

* Knowledge of a wide range of tools used in software development.
* An understanding of [Agile software development](https://www.codecademy.com/resources/blog/what-is-agile/?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=cw_what_is_devops_blog).
* Experience with data management.
* An understanding of cloud architecture, which is often used in DevOps.
* An ability to work as part of a team and lead a team as needed.
* Proficiency in automation.
* Proficiency with one or more [programming languages](https://www.codecademy.com/resources/blog/programming-languages/?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=cw_what_is_devops_blog).
* An understanding of [quality assurance](https://www.codecademy.com/resources/blog/what-is-a-qa-engineer/#what-is-quality-assurance?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=cw_what_is_devops_blog).
* An ability to troubleshoot.
* A willingness to talk with customers and incorporate their feedback.
* An understanding of security best practices.

**4.Implementation of CICD with Java and open source stack:-** Configure the web application and Version control using Git using Git commands and version control operations.

# **Git Commands**

There are many different ways to use Git. Git supports many command-line tools and graphical user interfaces. The Git command line is the only place where you can run all the Git commands.

The following set of commands will help you understand how to use Git via the command line.

## **Basic Git Commands**

Here is a list of most essential Git commands that are used daily.

1. [Git Config command](https://www.javatpoint.com/git-commands#config-command)
2. [Git init command](https://www.javatpoint.com/git-commands#init-command)
3. [Git clone command](https://www.javatpoint.com/git-commands#clone-command)
4. [Git add command](https://www.javatpoint.com/git-commands#add-command)
5. [Git commit command](https://www.javatpoint.com/git-commands#commit-command)
6. [Git status command](https://www.javatpoint.com/git-commands#status-command)
7. [Git push Command](https://www.javatpoint.com/git-commands#push-command)
8. [Git pull command](https://www.javatpoint.com/git-commands#pull-command)
9. [Git Branch Command](https://www.javatpoint.com/git-commands#branch-command)
10. [Git Merge Command](https://www.javatpoint.com/git-commands#merge-command)
11. [Git log command](https://www.javatpoint.com/git-commands#log-command)
12. [Git remote command](https://www.javatpoint.com/git-commands#remote-command)

## **1) Git config command**

This command configures the user. The Git config command is the first and necessary command used on the Git command line. This command sets the author name and email address to be used with your commits. Git config is also used in other scenarios.

**Syntax**

1. $ git config --global user.name "ImDwivedi1"
2. $ git config --global user.email "Himanshudubey481@gmail.com"

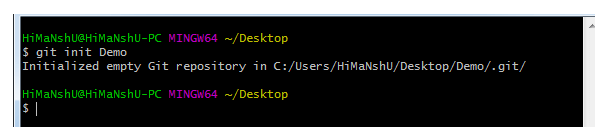
### 2) Git Init command

This command is used to create a local repository.

**Syntax**

1. $ git init Demo

The init command will initialize an empty repository. See the below screenshot.

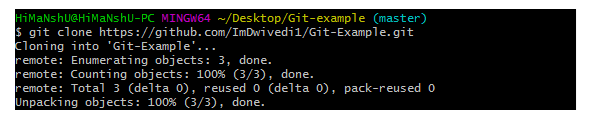


### 3) Git clone command

This command is used to make a copy of a repository from an existing URL. If I want a local copy of my repository from GitHub, this command allows creating a local copy of that repository on your local directory from the repository URL.

**Syntax**

1. $ git clone URL



### 4) Git add command

This command is used to add one or more files to staging (Index) area.

**Syntax**

To add one file

1. $ git add Filename

To add more than one file

1. $ git add\*

Git Commands

### 5) Git commit command

Commit command is used in two scenarios. They are as follows.

**Git commit -m**

his command changes the head. It records or snapshots the file permanently in the version history with a message.

**Syntax**

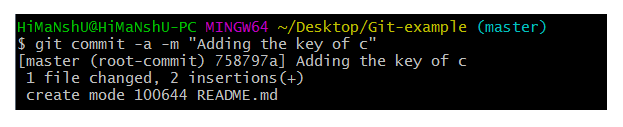
1. $ git commit -m " Commit Message"

**Git commit -a**

This command commits any files added in the repository with git add and also commits any files you've changed since then.

**Syntax**

1. $ git commit -a

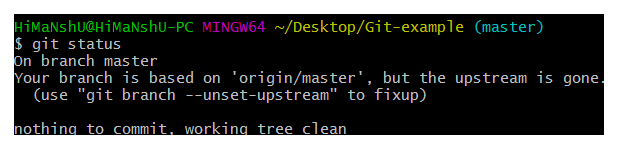


### 6) Git status command

The status command is used to display the state of the working directory and the staging area. It allows you to see which changes have been staged, which haven't, and which files aren?t being tracked by Git. It does not show you any information about the committed project history. For this, you need to use the git log. It also lists the files that you've changed and those you still need to add or commit.

**Syntax**

1. $ git status



### 7) Git push Command

It is used to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repo. It's the complement to git fetch, but whereas fetching imports commits to local branches on comparatively pushing exports commits to remote branches. Remote branches are configured by using the git remote command. Pushing is capable of overwriting changes, and caution should be taken when pushing.

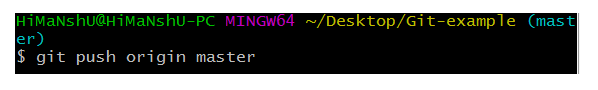
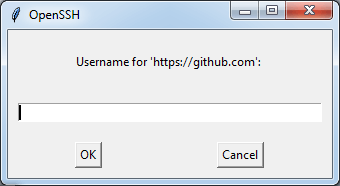
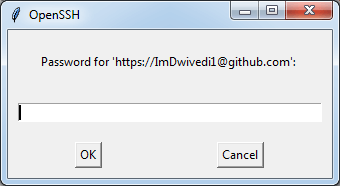
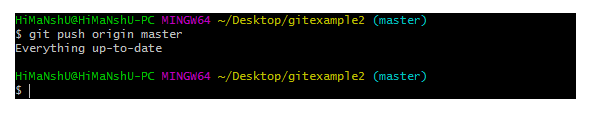
Git push command can be used as follows.

**Git push origin master**

This command sends the changes made on the master branch, to your remote repository.

**Syntax**

1. $ git push [variable name] master

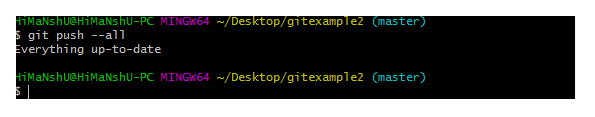
  
  
  


**Git push -all**

This command pushes all the branches to the server repository.

**Syntax**

1. $ git push --all

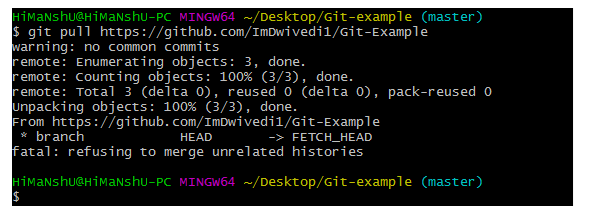


### 8) Git pull command

Pull command is used to receive data from GitHub. It fetches and merges changes on the remote server to your working directory.

**Syntax**

1. $ git pull URL

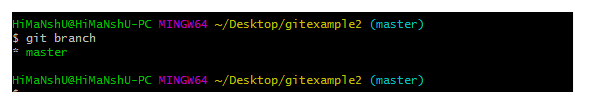


### 9) Git Branch Command

This command lists all the branches available in the repository.

**Syntax**

1. $ git branch



### 10) Git Merge Command

This command is used to merge the specified branch?s history into the current branch.

**Syntax**

1. $ git merge BranchName

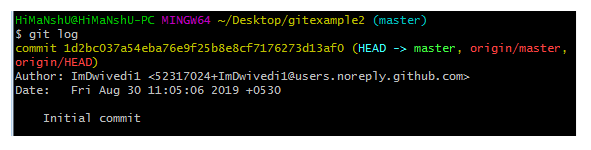


### 11) Git log Command

This command is used to check the commit history.

**Syntax**

1. $ git log



By default, if no argument passed, Git log shows the most recent commits first. We can limit the number of log entries displayed by passing a number as an option, such as -3 to show only the last three entries.

1. $ git log -3

### 12) Git remote Command

Git Remote command is used to connect your local repository to the remote server. This command allows you to create, view, and delete connections to other repositories. These connections are more like bookmarks rather than direct links into other repositories. This command doesn't provide real-time access to repositories.

