DevOps Assignment

P.VEERABABU
22K61A05D9
CSE-B

Set-1

1. **Explain importance of Agile software development.**

- The agile software development life cycle is a software development project methodology that prioritizes adaptability, flexibility, rapid development, and transformation.
- Agile Model is a combination of the Iterative and incremental model. This model focuses more on flexibility while developing a product rather than on the requirement.
- In Agile, a product is broken into small incremental builds. It is not developed as a complete product in one go. Each build increments in terms of features. The next build is built on the previous functionality.

- **Customer Focus**: Agile encourages frequent feedback from customers and stakeholders, ensuring that the product aligns closely with user needs. It allows for adjustments to be made early, which helps to avoid costly mistakes.
- **Flexibility & Adaptability**: Unlike traditional methods, Agile is designed to handle change. Requirements can evolve as the project progresses, allowing teams to adapt to new technologies, business priorities, or market shifts without major disruptions.
- **Faster Delivery**: Agile breaks the project into smaller, manageable iterations (sprints). This allows teams to deliver working software quickly and regularly, providing stakeholders with tangible results in shorter timeframes. It also helps prioritize critical features early in the process.
- **Improved Collaboration**: Agile promotes close collaboration between developers, designers, and business stakeholders. Daily standups, sprint reviews, and retrospectives help ensure that everyone is aligned and any issues can be addressed promptly.

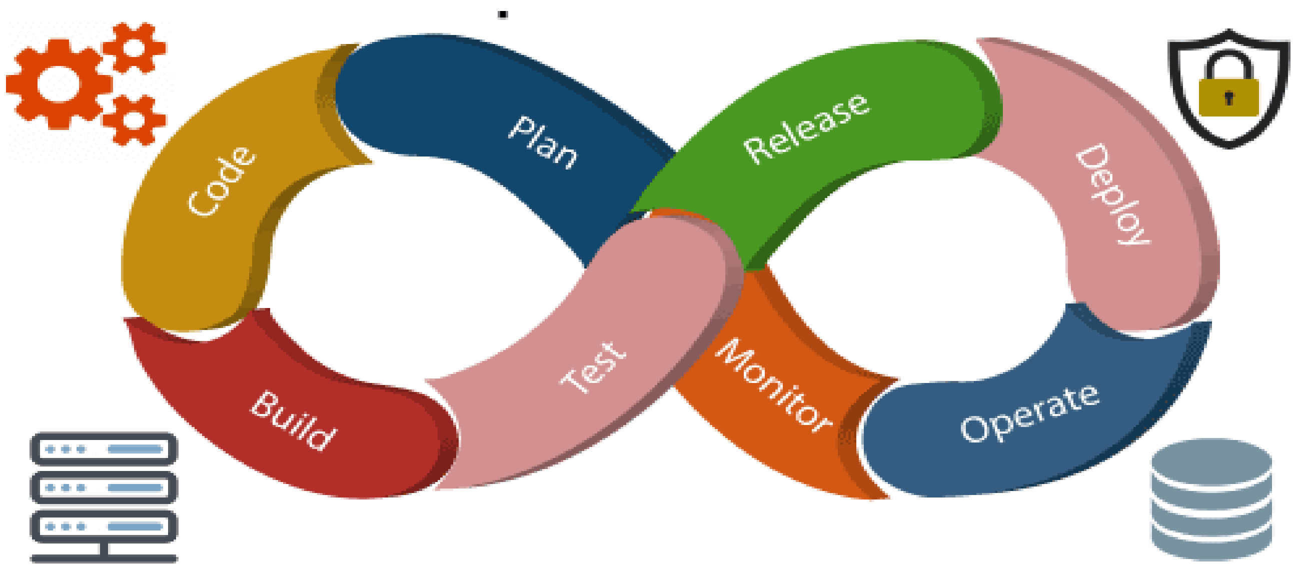
- **Higher Quality**: Agile emphasizes continuous testing, integration, and feedback. This leads to fewer bugs and defects, as quality is built into the process from the start rather than being a last-minute concern.
- **Transparency & Visibility**: With Agile, project progress is tracked transparently. Teams can see what has been done, what's next, and what's outstanding. This transparency keeps everyone informed and reduces misunderstandings.
- **Empowered Teams**: Agile encourages self-organizing teams, giving them the autonomy to make decisions and solve problems creatively. This boosts motivation and leads to more innovative solutions.
- **Risk Management**: By delivering work in small increments and getting constant feedback, Agile reduces the likelihood of major failures. If issues arise, they can be identified and resolved early, minimizing overall project risk.

2. Explain DevOps architecture and its features with a neat sketch.

- DevOps is a set of practices, principles, and cultural philosophies that emphasize collaboration and communication between software development (Dev) and IT operations (Ops) teams.

DevOps Architecture

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



- **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.

- **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.

- **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.

- **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.

- **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**.

- **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.

Benefits of DevOps Architecture

Faster Delivery: Automates repetitive tasks, reducing deployment time.

Improved Collaboration: Breaks silos between teams for better coordination.

Enhanced Reliability: Ensures consistent environments and robust monitoring.

Scalability: Enables efficient scaling of applications and infrastructure.

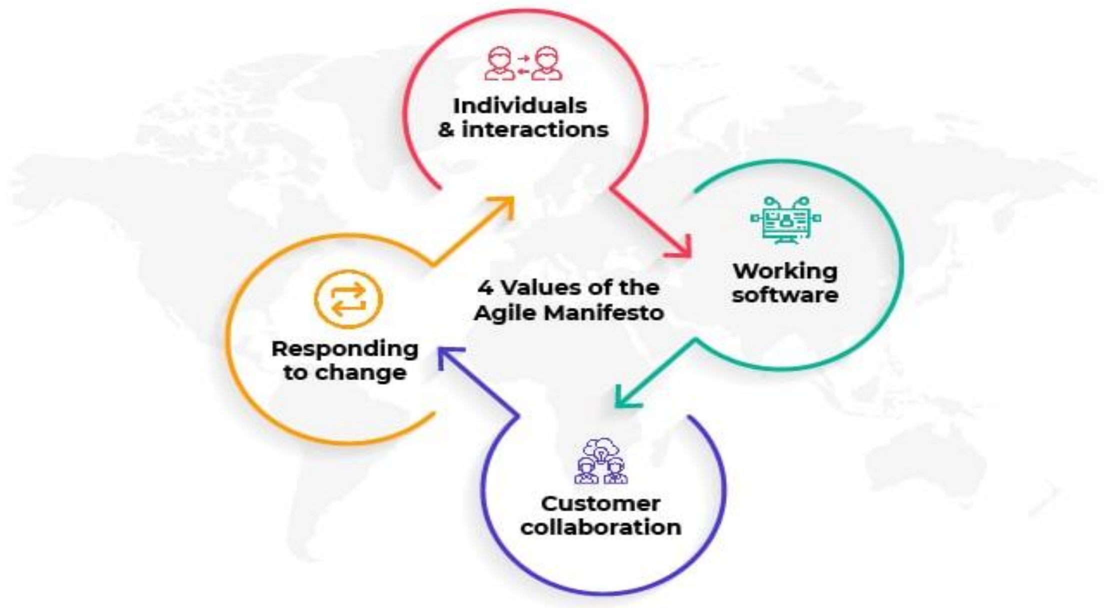
Cost Efficiency: Optimizes resource usage through automation and cloud-native solutions

3. Describe various features and capabilities in agile.

- Agile capabilities refer to an organization's ability to effectively adopt and implement agile methodologies and practices in their software development process.

- Agile is a mindset and a methodology that focuses on delivering value to customers through iterative and incremental development, with a focus on collaboration, flexibility, and continuous improvement.

- Agile capabilities are therefore critical to the success of any DevOps project, as DevOps also relies on a collaborative and iterative approach to software development.



- **Agile mindset**: It involves a willingness to embrace change, a focus on delivering value to customers, and a commitment to continuous improvement. The agile mindset is a culture that values teamwork, collaboration, and open communication. Organizations need to foster an agile mindset throughout their teams to effectively adopt DevOps practices.

- **Agile methodologies**: Agile methodologies like Scrum, Kanban, and XP provide a framework for agile development. These methodologies help teams to prioritize work, manage work in progress, and continuously improve the development process. Organizations need to select the appropriate agile methodology that aligns with their business needs and the nature of their project.

- **Agile practices**: Agile practices like user stories, sprint planning, and retrospectives help teams to collaborate, communicate, and deliver high-quality software. User stories help to capture requirements in a customer-centric way, while sprint planning helps to organize work into manageable chunks. Retrospectives provide an opportunity to reflect on the development process and make improvements.

- **Agile tools**: Agile tools like Jira, Trello, and GitLab can help teams to manage their work and collaborate effectively. These tools provide a platform for managing work in progress, tracking progress, and communicating with team members.

- **Continuous delivery**: Continuous delivery is an agile practice that involves continuously integrating and testing code changes, and deploying those changes to production quickly and frequently. This requires an agile approach to development, where teams work in small increments and focus on delivering value to customers.

- **Continuous improvement**: Continuous improvement is a core tenet of both agile and DevOps. Teams need to continuously reflect on their work and make improvements to their processes, tools, and practices. This involves an agile mindset, where teams are willing to experiment, learn from their mistakes, and make changes to improve the software delivery process.

SET-2

1. What is SDLC? Explain various phases involved in SDLC.

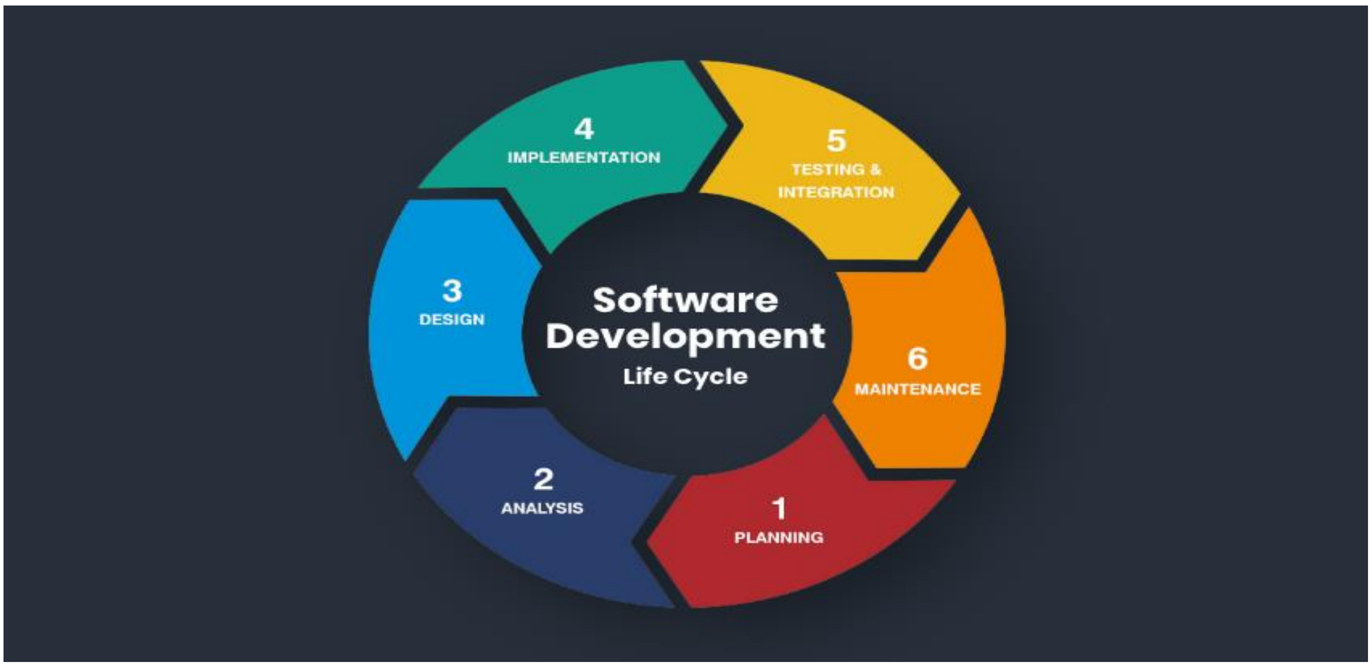
SDLC (Software Development Life Cycle):

- A framework that describes the activities performed at each stage of a software development project.
- A systematic approach that generates a structure for the developer to design, create and deliver high-quality software based on customer requirements and needs. The primary goal of the SDLC process is to produce cost-efficient and high-quality products. The process comprises a

detailed plan that describes how to develop, maintain, and replace the software.

Phases of SDLC

- Planning
- Analysis
- Design
- Implementation (Coding)
- Testing & Deployment
- Maintenance



Planning

- A feasibility study also takes place during the Planning phase. Developers and product teams evaluate technical and financial challenges that might affect the software's development or success.
- Key documents such as the Project Plan and Software Requirement Specification (SRS) are created.

- The Planning phase fosters effective communication and collaboration within the team. By defining clear roles, responsibilities, and expectations, it lays a solid foundation for an efficient software development process.

Analysis

- The development team will then analyze the requirements keeping the design and code of the software in mind. Further, investigating the validity and possibility of incorporating these requirements into the software system.
- The main goal of this stage is that everyone understands even the minute detail of the requirement. Hardware, operating systems, programming, and security are to name the few requirements.
- The project team collects information from stakeholders, including analysts, users, and clients. They conduct interviews, surveys, and focus groups to understand the user's expectations and needs. The process involves not only asking the right questions but also accurately interpreting the responses.

Design

- The Design phase is all about building the framework. Key activities include crafting data flow diagrams, constructing entity-relationship diagrams, and designing user interface mock-ups. The team also identifies system dependencies and integration points. They also set the software
- 's limitations, such as hardware constraints, performance requirements, and other system-related factors.
- The culmination of these tasks is an exhaustive Software Design Document (SDD). This document serves as the roadmap for the team during the coding phase. It meticulously details the software's design, from system architecture to data design, and even user interface specifics.

Coding

- The Coding phase in the Software Development Life Cycle (SDLC) is when engineers and developers get down to business and start converting the software design into tangible code.

- This development phase aims to develop software that is functional, efficient, and user-friendly. Developers use an appropriate programming language, Java or otherwise, to write the code, guided by the SDD and coding guidelines. This document, acting as a roadmap, ensures the software aligns with the vision set in earlier phases.

- Another key aspect of this phase is regular code reviews. Team members carefully examine each other's work to identify any bugs or inconsistencies.

Testing

- Once the developers build the software, then it is deployed in the testing environment. Then the testing team tests the functionality of the entire system. In this fifth phase of SDLC, the testing is done to ensure that the entire application works according to the customer's requirements.

- After testing, the QA and testing team might find some bugs or defects and communicate the same with the developers. The development team then fixes the bugs and send it to QA for a re-test. This process goes on until the software is stable, bug-free and working according to the business requirements of that system.

Deployment

- The sixth phase of SDLC: Once the testing is done, and the product is ready for deployment, it is released for customers to use. The size of the project determines the complexity of the deployment.

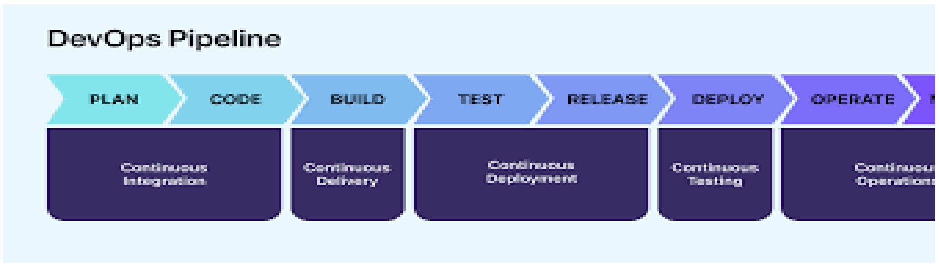
- The users are then provided with the training or documentation that will help them to operate the software. Again, a small round of testing is performed on production to ensure environmental issues or any impact of the new release.

Maintenance

- The maintenance phase is characterized by constant assistance and improvement, which guarantees the software's best possible functioning and longevity and ensures it meets customer expectations.

- The primary focus is to adapt to the software's changing needs. This adaptation involves responding to user feedback, resolving unexpected issues, and upgrading the software based on users' evolving requirements. It's a continuous process of refining and adapting.

2. Explain briefly about various stages involved in the DevOps pipeline.

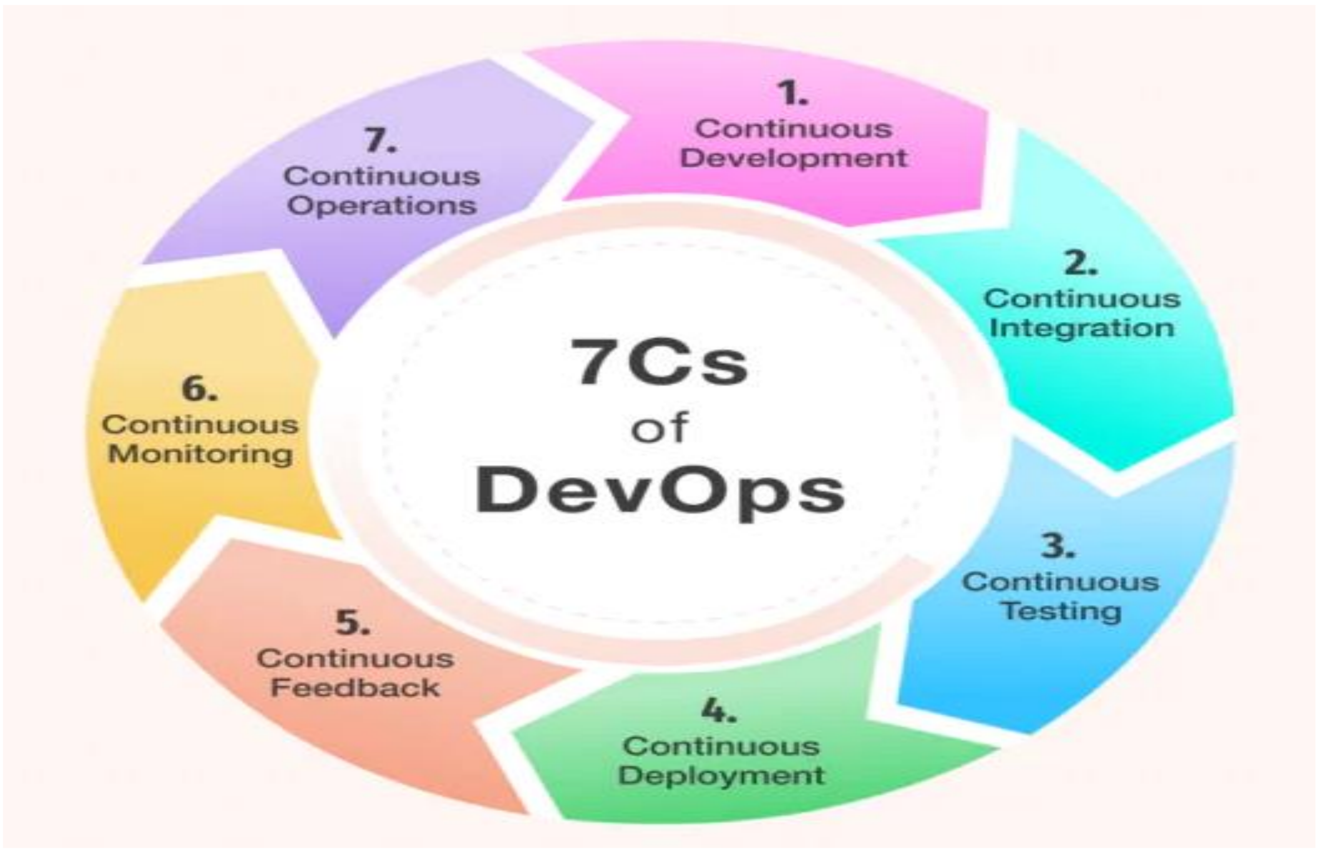


The DevOps pipeline involves several stages that help automate and streamline the process of software development, testing, and deployment. Here's a brief explanation of the key stages:

1. **Planning**: This is the initial stage where the requirements, features, and tasks are defined. It involves collaboration between development, operations, and stakeholders to ensure clear goals for the project.
2. **Coding**: In this stage, developers write the application's code. Tools like Git are commonly used for version control to manage the codebase.
3. **Building**: The code is compiled, packaged, and prepared for testing. The build process ensures that the code integrates properly and is ready for deployment.
4. **Testing**: Automated tests are run to check for bugs, security vulnerabilities, and performance issues. Continuous integration tools often trigger these tests whenever there are changes in the codebase.
5. **Release**: Once the code passes the tests, it’s released to staging environments for final validation. Release management tools may assist in automating this process.
6. **Deployment**: The code is deployed to production environments. In some cases, this is done using continuous deployment practices, ensuring faster delivery of features and fixes.

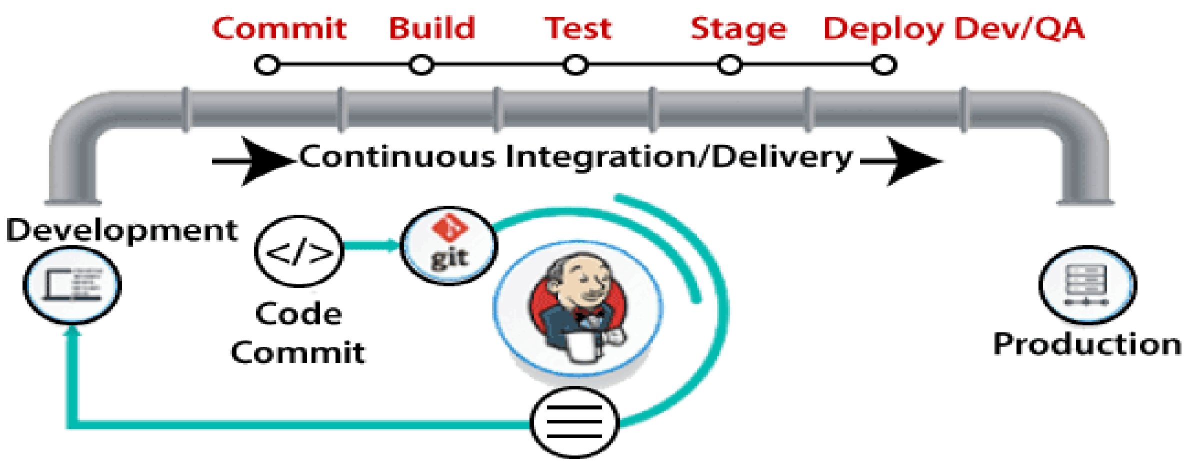
7. **Operations**: After deployment, the operations team monitors the application to ensure it runs smoothly. Tools like monitoring and logging are used to track performance and resolve issues proactively.
8. **Monitoring & Feedback**: Continuous monitoring provides feedback on the app's performance and user experience. This data informs future development and can lead to improvements, starting the cycle again.

3. **Describe the phases in DevOps life cycle.**



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. During this phase, project requirements are gathered and discussed with stakeholders. Moreover, the product backlog is also maintained based on customer feedback which is broken down into smaller releases and milestones for continuous software development.

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 not only involves 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.

3. **Continuous Testing:** This phase, where the developed software is continuously tested 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 be automated 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.

4. **Continuous Deployment:** Continuous deployment, or CD, is the final piece of a complete DevOps pipeline and automates the deployment of code releases.

   That means if code passes all automated tests throughout the production pipeline, it's immediately released to end users.

   CD critically removes the need for human intervention to orchestrate a software release, which results in faster release timelines. This also gives developers more immediate real-world feedback.

5. **Continuous Monitoring:** Continuous monitoring is a set of automated processes and tooling used to troubleshoot issues and development teams can use to inform future development cycles, fix bugs, and patch issues.

- A well established continuous monitoring system will typically contain four components:

  - **Logging** offers a continuous stream of raw data about business-critical components.

  - **Monitoring** provides intelligence about the raw data provided in logs and metrics.

  - **Alerting** provides proactive notifications when something has gone wrong and critical debugging information.

  - **Tracing** takes logging a step further, providing a deeper level of application performance and behavioral insights that can greatly impact the stability and scalability of applications in production environments.

6. **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.

SET-3

1. Write the difference between Waterfall and Agile models.

In the context of **DevOps**, the differences between **Waterfall** and **Agile** models can be summarized in the following table:

Aspect	Waterfall Model in DevOps	Agile Model in DevOps
Development Approach	Sequential and linear, with distinct phases (e.g., requirements, design, development, testing, deployment)	Iterative and incremental, with frequent releases and iterations
Release Cycle	Long release cycle, typically at the end of the project	Frequent and continuous releases, typically on a weekly or bi-weekly basis
Collaboration	Limited collaboration between development and operations teams during the project	Strong collaboration between development, operations, and other stakeholders throughout the project
Feedback	Feedback is collected late in the project, usually after deployment or during testing	Continuous feedback from all teams, stakeholders, and users, often during every iteration
Flexibility to Changes	Difficult to accommodate changes once development begins due to strict planning	Highly flexible; frequent iterations allow for changes and adjustments based on feedback
Testing &	Testing and deployment	Continuous testing and

Deployment	occur after development is complete	integration, leading to frequent deployments
Risk Management	Risk is identified and addressed later, leading to potential delays or issues	Risk is continuously managed, with regular iterations to identify and address issues early
Automation	Limited automation, as tasks and processes follow a set sequence	High emphasis on automation (CI/CD pipelines, automated testing, deployment) to enable faster delivery
Customer Involvement	Customer is involved mainly in initial phases and at the end of the project	Regular customer involvement and feedback throughout the entire project life cycle
Change Management	Change management is rigid, and changes are costly once the project has started	Change management is flexible and ongoing, allowing for adjustments after each iteration
Speed of Delivery	Slow delivery, with updates or releases only at the end of the project	Fast delivery, with incremental updates and releases happening regularly

In **DevOps**, the Agile model is often preferred due to its focus on collaboration, automation, continuous delivery, and the ability to handle frequent changes, which aligns well with DevOps principles of faster, more efficient software development and delivery. The Waterfall model, on the other hand, is more rigid and less adaptable to the continuous, iterative nature of DevOps.

2.Discuss in detail about DevOps eco system.

DevOps Architecture:

The **DevOps ecosystem** refers to a set of practices, tools, and cultural philosophies that help organizations improve collaboration and communication between software development (Dev) and IT operations (Ops) teams.

Key Components of the DevOps Ecosystem

1. **Collaboration and Culture**:
   - **DevOps Culture**: At the heart of DevOps is a shift in culture. The goal is to break down silos between development, operations, quality assurance, and other departments to foster collaboration, shared responsibility, and faster feedback loops. Teams work together through every stage of the development lifecycle, from planning to release and beyond.
   - **Communication**: In a DevOps ecosystem, communication between all teams (development, QA, operations, security) is essential. Collaboration tools like Slack, Microsoft Teams, and Jira help facilitate this communication and keep everyone in sync.
2. **Continuous Integration (CI)**:
   - **Definition**: Continuous Integration is the practice of frequently integrating code changes into a shared repository. Developers regularly commit their code to a central version control system (VCS), where automated builds and tests are triggered to ensure the new code doesn't break existing functionality.
   - **Tools for CI**:
     - **Jenkins**
     - **CircleCI**
     - **Travis CI**
     - **GitLab CI**
3. **Continuous Delivery (CD)**:

- **Definition**: Continuous Delivery builds on CI by automating the process of deploying code to different environments (e.g., staging, production) so that software can be released at any time. The goal of CD is to ensure that the code is always in a deployable state.
- **Tools for CD**:
  - **Jenkins**
  - **Spinnaker**
  - **Octopus Deploy**

4. **Infrastructure as Code (IaC)**:
   - **Definition**: Infrastructure as Code refers to managing and provisioning computing infrastructure through machine-readable script files rather than manual processes. IaC allows teams to automate the creation, configuration, and management of infrastructure, making it consistent, repeatable, and scalable.
   - **Tools for IaC**:
     - **Terraform**
     - **Ansible**
     - **Chef**
     - **Puppet**
5. **Version Control System (VCS)**:
   - **Definition**: Version control is the process of tracking and managing changes to code over time. A central repository allows multiple developers to collaborate on the same codebase.
   - **Tools for VCS**:
     - **Git**
     - **GitHub**
     - **GitLab**
     - **Bitbucket**

6. **Automated Testing**:
    - **Definition**: Automated testing is the practice of using software tools to automatically test code for defects before it is released. It helps ensure that code is of high quality, functions correctly, and doesn't introduce bugs when changes are made.
    - **Tools for Automated Testing**:
        - **JUnit**
        - **Selenium**
        - **JUnit, TestNG**
        - **Postman**
7. **Monitoring and Logging**:
    - **Definition**: Continuous monitoring and logging of applications and infrastructure are critical to DevOps practices. It enables teams to detect issues quickly and ensure that systems run as expected. Logs provide insights into system health, and monitoring tools provide real-time alerts on system performance.
    - **Tools for Monitoring and Logging**:
        - **Prometheus**
        - **Grafana**
        - **Nagios**
        - **Splunk**
        - **Datadog**
8. **Containerization**:
    - **Definition**: Containers allow developers to package and isolate applications and their dependencies into a portable and reproducible unit. This ensures that software runs consistently across different environments.
    - **Tools for Containerization**:

        - **Docker**
        - **Kubernetes**
        - **OpenShift**

9. **Cloud Computing**:
    - **Definition**: Cloud platforms enable teams to host and scale applications in a flexible, on-demand manner. DevOps leverages cloud services to automate the provisioning of infrastructure and applications without managing physical hardware.
    - **Cloud Providers**:
        - **Amazon Web Services (AWS**
        - **Microsoft Azure**
        - **Google Cloud Platform (GCP**
        - **IBM Cloud**
10. **Security (DevSecOps)**:
    - **Definition**: DevSecOps refers to integrating security practices within the DevOps pipeline. It emphasizes proactive security measures at every stage of the software development lifecycle.
    - **Tools for DevSecOps**:
        - **SonarQube**
        - **Snyk**
        - **OWASP ZAP**
        -

3.List and explain the steps followed for adopting DevOps in IT projects.

- DevOps is a set of practices that brings together software development and IT operations to enable the continuous delivery of high-quality software.

- Adopting DevOps practices in projects can have several benefits, including **faster delivery of software**, **improved quality**, and **increased collaboration** between development and operations teams.
- To adopt DevOps practices in projects, organizations need to follow a few key steps:

- **Cultural shift**: Adopting DevOps requires a cultural shift in the organization. This means breaking down silos and creating a culture of collaboration, communication, and continuous improvement.
- **Tools and automation**: DevOps requires tools and automation to enable continuous integration, delivery, and deployment. This includes tools for source code management, build automation, testing, and deployment automation.

- **Continuous testing**: DevOps emphasizes continuous testing throughout the software development life cycle to ensure that code is delivered with high quality and is free of defects.
- **Continuous monitoring**: DevOps requires continuous monitoring of the software in production to identify issues and improve performance.

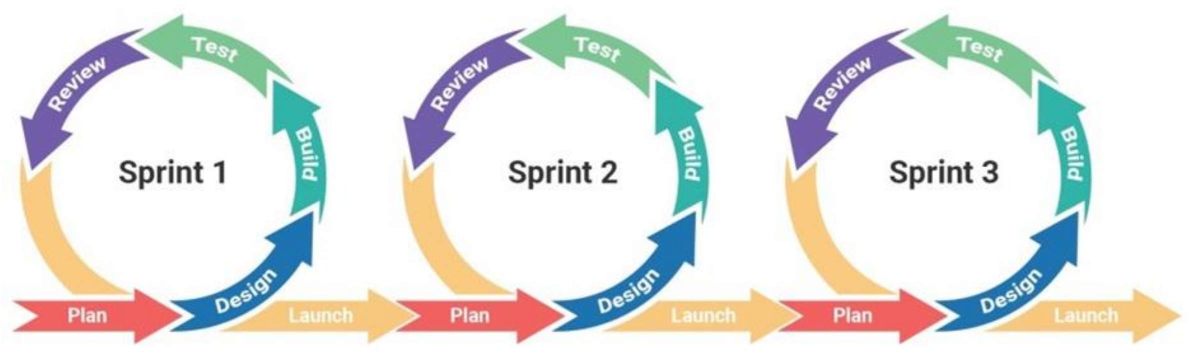
- **Feedback loops**: DevOps requires feedback loops to enable continuous improvement. This includes feedback loops between development and operations teams, as well as between the software and its users.
- Overall, the adoption of DevOps practices requires a commitment to continuous improvement and a willingness to change the way that software is developed and delivered.

- It can be challenging to implement, but the benefits can be significant, including faster delivery of high-quality software and increased collaboration between teams.

SET-4

1.Explain the values and principles of Agile model.

- The agile model in DevOps is an iterative approach to software development that involves breaking down projects into smaller parts and delivering them in cycles.



Agile Values

- Individuals and interactions

  Along with the software tools and processes, agile suggests that the people working in the processes are equally if not more important. A project can be the most successful if the people working on it are the best suited for it.
- A working software

  Working software is more important. It is of utmost importance according to the agile manifesto to provide the customers with working software than to have piles and piles of documentation.

Agile Principles

- Satisfy the customer with recurring delivery of software at regular intervals.
- If there are changes in requirements then accept them, even if they arrive late in the development cycle
- Deliver the software at intervals between 3 weeks to 3 months.
- The developers and the businessmen should work in collaboration.
- Building projects keeping in mind individuals who are highly motivated and then

supporting them to get the work done.

- Using face to face conversation as the most effective way of communication.
- To measure progress the software should be working.
- The development should be sustainable and there should be consistency.
- Technical excellence and a good design help in better agility.
- Simplicity is the key.
- A team which is self-organizing helps to deliver best architecture, design and requirements.
- The team decides and discusses ways in which the team can be more effective.

2.Write a short notes on the DevOps Orchestration.

DevOps Orchestration

 DevOps automation is a process by which a single, repeatable task, such as launching an app or changing a database entry, is made capable of running without human intervention, both on PCs and in the cloud.

 Orchestration refers to a set of automated tasks that are built into a single workflow to solve a group of functions such as managing containers, launching a new web server, changing a database entry, and integrating a web application. More simply, orchestration helps configure, manage, and coordinate the infrastructure requirements an application needs to run effectively.

 Automation applies to functions that are common to one area, such as launching a web server, or integrating a web app, or changing a database entry. But when all of these functions must work together, DevOps orchestration is required.

 DevOps orchestration involves automating multiple processes to reduce issues leading to the production date and shorten time to market. On the other hand, automation is used to perform tasks or a series of actions that are repetitive.

 DevOps orchestration streamlines the entire workflow by centralizing all tools used across teams, along with their data, to keep track of process and completion status throughout. Besides, automation can be pretty complicated at scale, although normally it is focused on a specific operation to achieve a goal, such as a server deployment. When automation has reached its limitations, that's when orchestration plays to its strengths.

3.What is the difference between Agile and DevOps models?

Agile: Agile program advancement comprises different approaches to computer program improvement beneath which prerequisites and arrangements advance through the collaborative exertion of self-organizing and cross-functional groups and their customer/end client.

DevOps: DevOps could be a set of hones that combines program improvement and information-technology operations which points to abbreviating the framework's advancement life cycle and giving nonstop conveyance with tall program quality.

Difference between Agile and DevOps:

S. No.	**Agile**	**DevOps**
1.	It started in the year 2001.	It started in the year 2007.
2.	Invented by John Kern, and Martin Fowler.	Invented by John Allspaw and Paul Hammond at Flickr, and the Phoenix Project by Gene Kim.
3.	Agile is a method for creating software.	It is not related to software development. Instead, the software that is used by DevOps is pre-built, dependable, and simple to deploy.
4.	An advancement and administration approach.	Typically a conclusion of administration related to designing.
5.	The agile handle centers on consistent changes.	DevOps centers on steady testing and conveyance.
6.	A few of the finest steps embraced in Agile are recorded underneath – 1. Backlog	DevOps to have a few best hones that ease the method – 1. Focus

S. No.	Agile	DevOps
	Building 2.Sprint advancement	on specialized greatness. 2. Collaborate straightforwardly with clients and join their feedback.
7.	Agile relates generally to the way advancement is carried of, any division of the company can be spry in its hones. This may be accomplished through preparation.	DevOps centers more on program arrangement choosing the foremost dependable and most secure course.
8.	All the group individuals working in a spry hone have a wide assortment of comparable ability sets. This is often one of the points of interest of having such a group since within the time of requirement any of the group individuals can loan help instead of holding up for the group leads or any pro impedances.	DevOps features a diverse approach and is very viable, most of the time it takes after “Divide and Conquer”. Work partitioned among the improvement and operation groups.
9.	Spry accepts “smaller and concise”. Littler the group superior it would be to convey with fewer complexities.	DevOps, on the other hand, accepts that “bigger is better”.
10.	Since Agile groups are brief, a foreordained sum of time is there which are sprints. Tough, it happens that a sprint has endured longer than a month but regularly a week long.	DevOps, on the other hand, prioritizes reliabilities. It is since of this behavior that they can center on a long-term plan that minimizes commerce’s unsettling influences.
11.	A big team for your project is not required.	It demands collaboration among different teams for the completion of work.
12.	**Some of the Tools-** • Bugzilla • JIRA • Kanboard and more.	**Some of the Tools-** • Puppet • Ansible • AWS • Chef • team City OpenStack and more.
13.	It is suitable for managing complex projects in any department.	It centers on the complete engineering process.
14.	It does not focus on the automation.	It focusses on automation.

S. No.	Agile	DevOps
15.	Working system gets more significance in Agile than documentation.	The process documentation is significant in DevOps.