DevOps Assignment

SET-1 :

1.Explain importance of Agile software development.

Importance of Agile Software Development :

Agile software development is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and customer feedback. The Agile methodology contrasts with traditional approaches, such as Waterfall, by breaking down large projects into smaller, more manageable units of work (called iterations or sprints).

Below is a detailed explanation of the importance of Agile software development:

1. **Customer-Centric Focus**: Agile emphasizes continuous collaboration with the customer, ensuring that their feedback is integrated throughout the development process. This leads to products that more accurately meet user needs and expectations.
2. **Adaptability and Flexibility**: One of the core principles of Agile is responding to change. As business or technical requirements evolve, Agile teams can quickly adjust their priorities, ensuring that the project stays aligned with real-world changes.
3. **Faster Time to Market**: Agile promotes iterative development with short release cycles (called sprints). This allows teams to deliver working software early and often, providing stakeholders with valuable product features sooner, which can help in capturing market opportunities quicker.
4. **Improved Collaboration and Communication**: Agile fosters a highly collaborative environment, where team members, stakeholders, and customers regularly communicate. This leads to better alignment, fewer misunderstandings, and a more cohesive project vision.
5. **Increased Quality**: Frequent testing, feedback, and iteration improve the quality of the product. Agile encourages teams to test early and often, reducing the likelihood of major issues later in development.
6. **Risk Reduction**: Because the product is developed in smaller, manageable chunks, teams can identify and address risks earlier, before they become major problems. Regular reviews also ensure that any issues are detected and fixed promptly.
7. **Enhanced Team Morale**: Agile promotes self-organizing teams with high autonomy and accountability, which increases job satisfaction and

productivity. Teams can work in a more collaborative and supportive environment, making work more enjoyable.

8. **Continuous Improvement**: Agile encourages teams to reflect regularly on their processes and make improvements. This allows for constant refinement of work practices, leading to greater efficiency and effectiveness over time.
9. **Cost Efficiency:**

Agile's incremental approach allows for better management of budgets. The flexibility in Agile means that resources are allocated effectively**, and if necessary, the project can be scaled down without significant cost implications. This** cost-conscious approach **makes Agile particularly attractive for businesses with tight budgets or in fast-changing markets.**

10. Improved Team Morale and Productivity :

Agile fosters an **empowered, self-organized team** environment. Developers are given the autonomy to make decisions and are encouraged to collaborate closely, which increases **ownership and accountability**

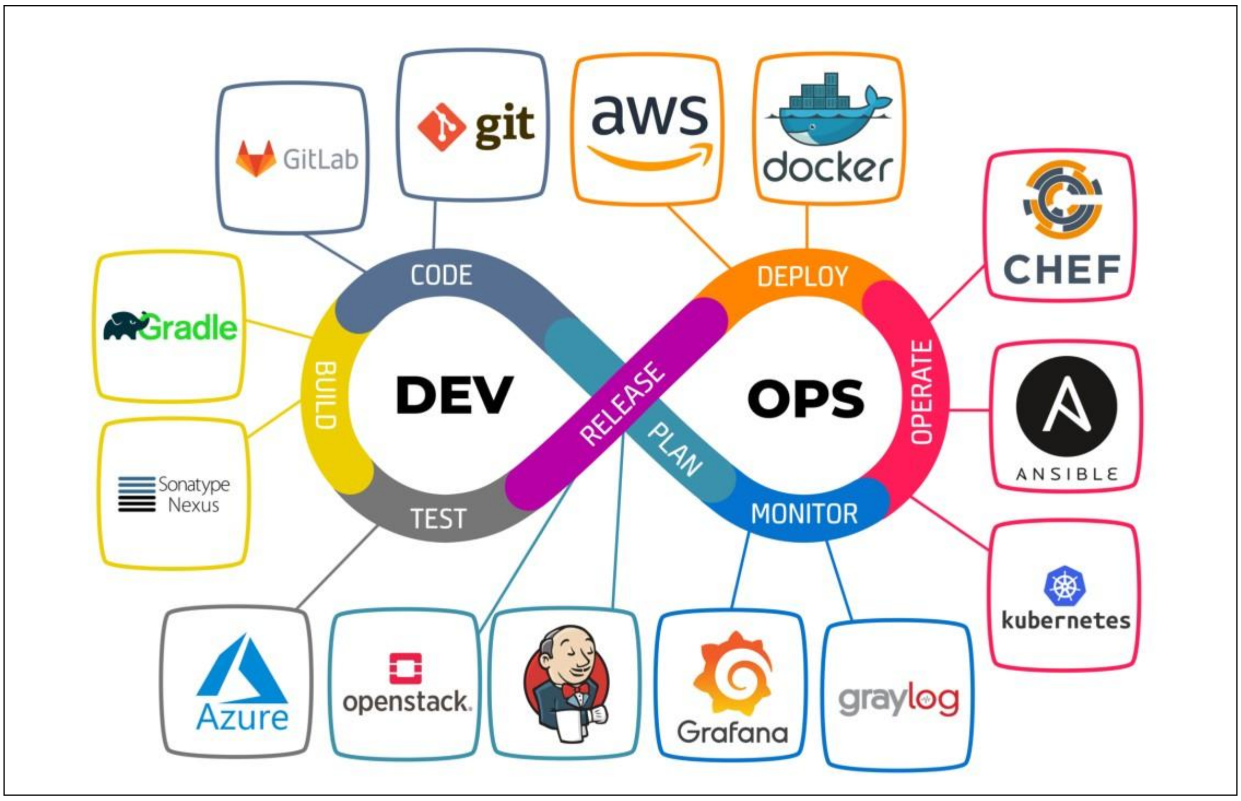
Conclusion

Agile software development is crucial for modern organizations that need to deliver high-quality products quickly, adapt to changing requirements, and maintain a competitive edge. Its focus on customer collaboration, flexibility, and iterative development allows for faster releases, better product quality, improved risk management, and more engaged teams. As businesses continue to operate in rapidly evolving environments, Agile has proven to be a powerful methodology that maximizes efficiency, collaboration, and innovation.

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

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.

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 teams; therefore, delivery can be faster.



DevOps Architecture

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.

Release:

Once tests pass, the application is ready for deployment. In Continuous Delivery, it goes to staging; in Continuous Deployment, it can go directly to production.

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.

Operate:

Once in production, the system is continuously monitored for performance, errors, and security issues.

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.

3. Describe various features and capabilities in agile.

The terms **features** and **capabilities** are often used interchangeably but have distinct meanings, especially in the context of Agile or software development. Here's a breakdown of the differences between the two:

1. Features

- **Definition**: A **feature** refers to a specific functionality or characteristic of a product, service, or system. Features are tangible and often describe what a product does or what it offers to its users.
- **Characteristics**:
  - **Descriptive**: Features typically describe what something can do, i.e., the attributes or aspects of the system.
  - **User-Centric**: Features are often based on the needs of the users and the business goals.
  - **Functionality**: They define the capabilities that the system provides to users or other systems.
- **Examples of Features in Agile**:
  - **Sprint Planning**: A process in which the team decides what tasks or user stories to work on in the next sprint.
  - **Customer Collaboration**: The feature that emphasizes regular and continuous feedback from customers during the development cycle.
  - **Frequent Deliverables**: The feature where working software is delivered at the end of each sprint, enabling quick feedback and iterations.

---

2. Capabilities

- **Definition**: A **capability** refers to the ability or potential of a system, team, or process to perform a specific action or achieve a specific goal. In Agile, capabilities describe how the features can be executed or leveraged to accomplish desired outcomes.
- **Characteristics**:
  - **Action-Oriented**: Capabilities focus on what the system, team, or process is **able to do**, rather than what it has or offers.
  - **Skills and Expertise**: Capabilities often involve the collective skills, knowledge, or capacity of a team to deliver the expected features and work.

    - **Enabling Functionality**: Capabilities describe the capacity to perform a task or achieve a goal, including the potential to improve and adapt over time.
- **Examples of Capabilities in Agile**:
    - **Adaptability and Flexibility**: The capability of an Agile team to quickly adjust to changing requirements or new business needs.
    - **Continuous Improvement**: The capability of a team to consistently reflect on and enhance its processes, leading to better performance over time.
    - **Collaboration**: The capability of a team to work together effectively, integrating both developers and stakeholders to deliver high-quality software.

SET-2 :

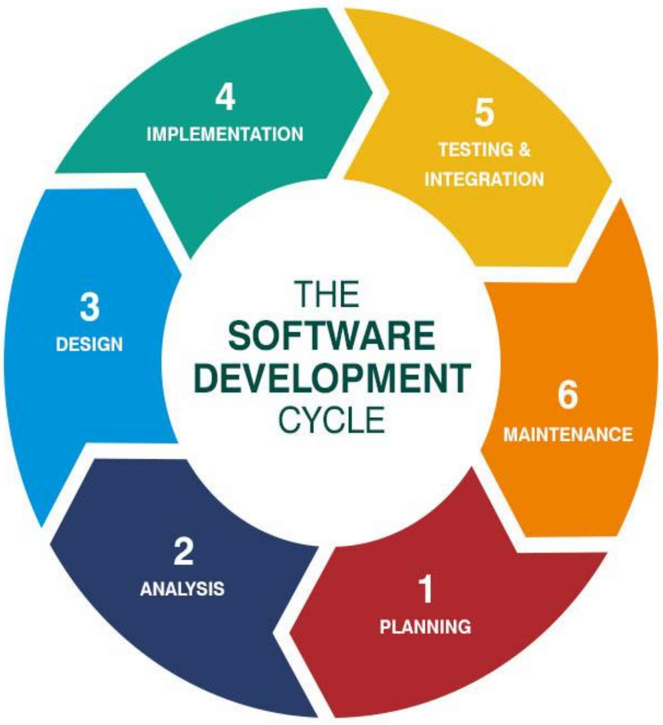
1. What is SDLC? Explain various phases involved in SDLC.

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

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

1.Planning
2.Analysis
3.Design
4.Implementation (Coding)
5.Testing & Deployment
6.Maintenance

Planning :

- Planning, involves defining the software's purpose and scope. The team collaborates to understand the end-users' needs and the goals the software should meet. Essentially, we ask, "What problem will this software solve?" and "What value will it offer to the user?".
- 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 second step of SDLC is gathering maximum information from the client requirements for the product. Discuss each detail and specification of the product with the customer.
- 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.

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.

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.

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.

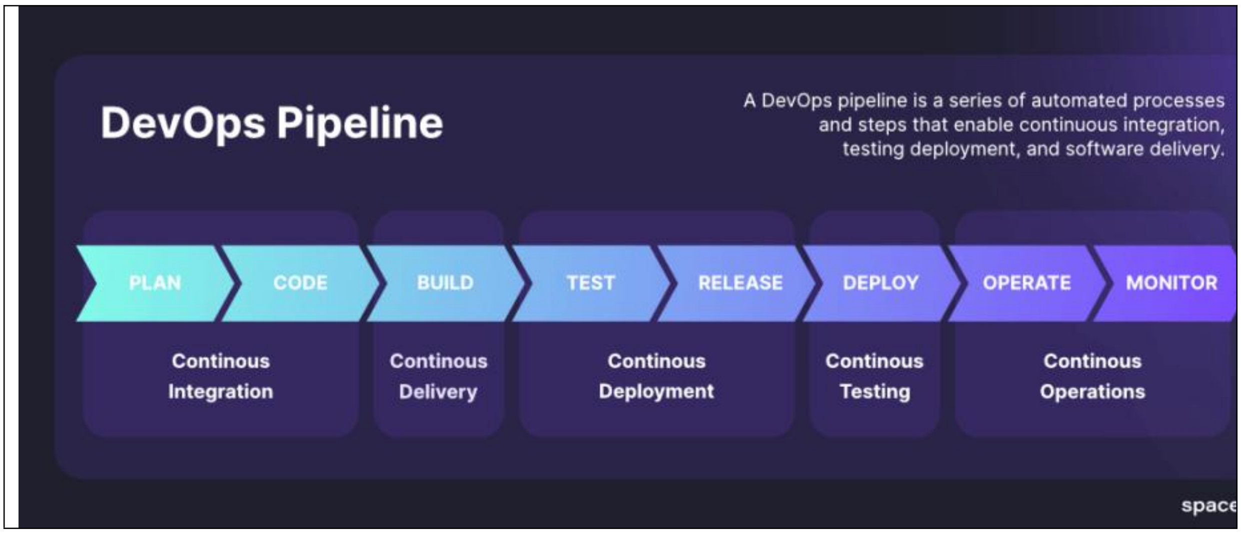
Deployment :

- 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 thecomplexity of the deployment.

Maintenance :

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

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



Plan :

- The Plan stage covers everything that happens before the developers start writing code, and it's where a Product Manager or Project Manager earns their keep. Requirements and feedback are gathered from stakeholders and customers and used to build a product roadmap to guide future development.

Code :

- This helps to teach developers good coding practice while aiding collaboration by providing some consistency to the codebase. These tools also help resolve issues that may fail tests later in the pipeline, resulting in fewer failed builds.

Build :

- The Build phase is where DevOps really kicks in. Once a developer has finished a task, they commit their code to a shared code repository.

- There are many ways this can be done, but typically the developer submits a pull request — a request to merge their new code with the shared codebase.

Test :

- Once the application is deployed to the test environment, a series of manual and automated tests are performed. At the same time, automated tests might run security scanning against the application, check for changes to the infrastructure and compliance with hardening best-practices, test the performance of the application or run load testing.
- The testing that is performed during this phase is up to the organisation and what is relevant to the application, but this stage can be considered a test-bed that lets you plug in new testing without interrupting the flow of developers or impacting the production environment.

Release :

- The Release phase is a milestone in a DevOps pipeline — it's the point at which we say a build is ready for deployment into the production environment. By this stage, each code change has passed a series of manual and automated tests, and the operations team can be confident that breaking issues and regressions are unlikely.

Deploy :

- Finally, a build is ready for the big time and it is released into production. There are several tools and processes that can automate the release process to make releases reliable with no outage window.
- The same Infrastructure-as-Code that built the test environment can be configured to build the production environment.

Operate :

- The new release is now live and being used by the customers. Geat work!
- The operations team is now hard at work, making sure that everything is running smoothly. Based on the configuration of the hosting service, the environment automatically scales with load to handle peaks and troughs in the number of active users.

Monitor :

- The ‘final’ phase of the DevOps cycle is to monitor the environment. this builds on the customer feedback provided in the Operate phase by collecting data and providing analytics on customer behaviour, performance, errors and more.
- There is no start or end, just the continuous evolution of a product throughout its lifespan, which only ends when people move on or don’t need it any more.

Continuous Integration :

One of the biggest difficulties in coordinating a software development team is managing the collaboration of many developers, often in remote locations, on a single codebase. A shared code repository is key to solving this problem, however, there can still be issues in when merging the changes made by multiple people on the same piece of code.

Continuous Delivery :

Continuous Delivery is an extension of Continuous Integration which automates the process of deploying a new build into production. Continuous Delivery aligns with the Test and Release phases of the pipeline, and allows organisations to manually trigger the release of new builds as regularly as they choose.

Continuous Deployment :

Continuous Deployment is a more advanced version of Continous Delivery (which makes the reuse of the ‘CD’ abbreviation more acceptable). The goals are the same, but the manual step of approving new releases into production is removed. In a Continuous Deployment model, each build which passes all of the checks and balances of the pipeline are automatically deployed into production.

Continuous Testing (CT)

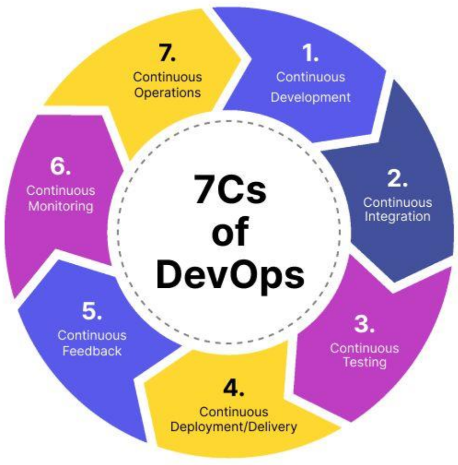
Continuous Testing refers to the practice of running automated tests continuously throughout the software development lifecycle (SDLC) to ensure that the codebase is always in a testable state. It involves integrating testing into every phase of development to detect issues early and provide real-time feedback to developers.

Continuous Operations (CO)

Continuous Operations refers to the practices and strategies that ensure an organization's IT systems, applications, and infrastructure remain up and running without interruption. The goal of Continuous Operations is to maintain business operations without downtime or disruption, providing uninterrupted services to users and customers.

3.Describe the phases in DevOps life cycle.

DevOps is a collaboration between Development and IT Operations to make software production and Deployment in an automated & repeatable way.



DevOps Lifecycle

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.**
- **Once the team agrees upon the business needs, the development team starts coding for the desired requirements. It’s a continuous process where developers are required to code whenever any changes occur in the project requirement or in case of any performance issues.**

- There are no DevOps tools that are required for planning, but there are several tools for maintaining the code.

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.
- In this phase, updated code or add-on functionalities and features are developed and integrated into existing code. Furthermore, bugs are detected and identified in the code during this phase at every step through unit testing, and then the source code is modified accordingly.
- This step makes integration a continuous approach where code is tested at every commit.
- 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.

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. 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.

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.

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 behavioural insights that can greatly impact the stability and scalability of applications in production environments.

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.

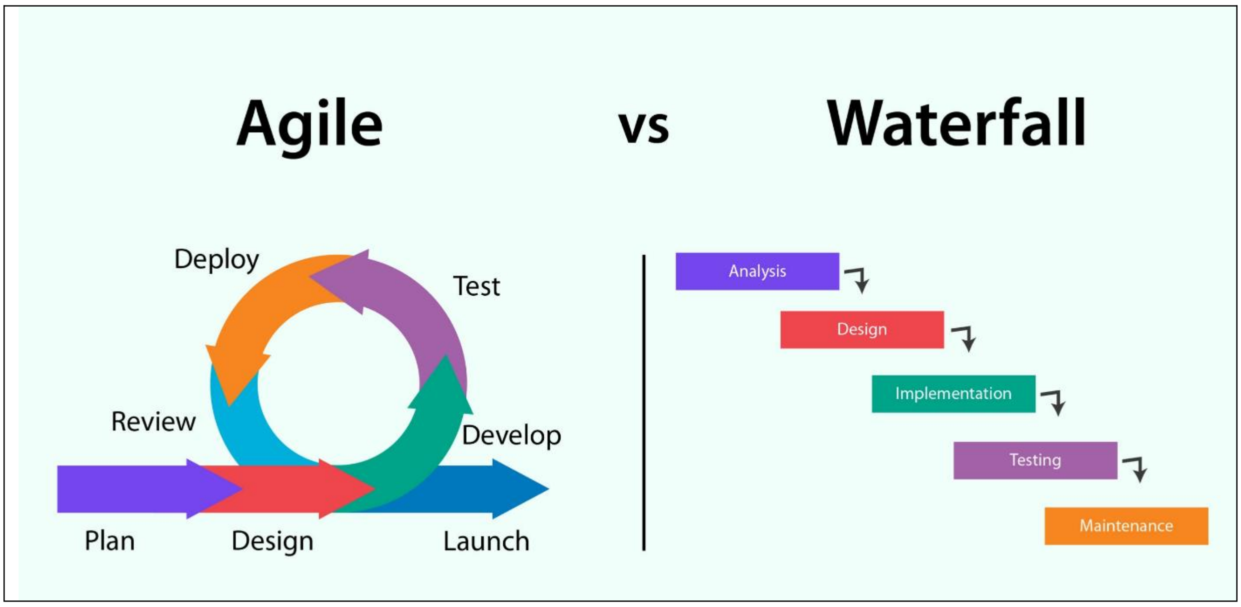
SET-3

1.Write the difference between Waterfall and Agile models.

Agile Methodology vs Waterfall Methodology :

Agile Methodology	Waterfall Methodology
Client input is required throughout the product development.	Client input is required only after completing each phase.
Changes can be made at any stage.	Changes cannot be made after the completion of a phase.
Coordination among project teams is required to ensure correctness.	Coordination is not needed as one team starts the work after the finish of another team.
It is really useful in large and complex projects.	It is mainly used for small project development.
The testing part can be started before the development of the entire product.	Testing can only be performed when the complete product is ready.
A Small team is sufficient for	It requires a large team.

Agile Methodology	Waterfall Methodology
Agile project management.	
The cost of development is less.	The cost of development is high.
It completes the project in comparatively less time.	It takes more time compared to Agile.
The Agile Method is known for its flexibility.	The waterfall Method is a structured software development methodology so it is quite rigid.
After each sprint/cycle test plan is discussed.	Hardly any test plan is discussed during a cycle.



2.Discuss in detail about DevOps eco system.

DevOps eco system :

A DevOps ecosystem refers to a collaborative environment where development and operations teams work closely together, utilizing a set of practices, tools, and cultural philosophies to automate the software development lifecycle, enabling faster delivery of applications with high quality, by integrating continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring throughout the process, fostering a culture of feedback and rapid iteration.

Key components of a DevOps ecosystem:

- **Continuous Development (CD):**The ongoing process of writing, reviewing, and committing code to a shared repository, ensuring frequent updates and collaboration between developers.
- **Continuous Integration (CI):** Regularly merging code changes from different developers into a central repository, automatically triggering builds and tests to identify integration issues early.
- **Continuous Testing (CT):** Automating various testing phases (unit, integration, system, acceptance) to quickly identify bugs and ensure code quality throughout the development cycle.
- **Continuous Deployment (CD):** Automatically deploying tested code to production environments, enabling rapid releases and frequent updates.
- **Continuous Monitoring (CM):** Real-time monitoring of application performance, infrastructure health, and user experience to detect potential issues and proactively address them.

Essential Tools in a DevOps Ecosystem:

- **Version Control Systems (VCS):** Like Git, used to manage code changes, track revisions, and facilitate collaboration.
- **CI/CD Pipelines:** Automated workflows that build, test, and deploy applications, often implemented with tools like Jenkins, CircleCI, or GitHub Actions.
- **Infrastructure as Code (IaC):** Managing infrastructure (servers, networks) through code, allowing for consistent deployments and easy scaling using tools like Terraform or AWS CloudFormation.
- **Containerization Platforms:** Docker and Kubernetes, enabling packaging applications with dependencies into self-contained containers for efficient deployment and scaling.

- **Monitoring Tools:** New Relic, Prometheus, Datadog, for real-time application performance monitoring and alerting.
- **Logging and Analytics Platforms:** ELK Stack (Elasticsearch, Logstash, Kibana) to collect, analyze, and visualize logs for troubleshooting and insights.

Cultural Aspects of DevOps:

- **Collaboration:** Breaking down silos between development and operations teams, promoting open communication and shared responsibility.
- **Feedback Loop:** Continuously gathering feedback from users and operations teams to improve the product and development process.
- **Automation:** Embracing automation to streamline repetitive tasks and reduce manual intervention.
- **Shared Ownership:** Developers taking ownership of the application throughout its lifecycle, including deployment and monitoring.

Benefits of a DevOps Ecosystem:

- **Faster Time to Market:** Rapidly delivering new features and updates to customers.
- **Improved Quality:** Early detection and resolution of bugs through continuous testing and integration.
- **Increased Reliability:** Proactive identification and mitigation of issues through monitoring.
- **Enhanced Efficiency:** Automation of routine tasks, freeing up teams to focus on innovation.

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

DevOps adoption in 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.

Agile model:

- 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.

Principles of Agile:

- 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.

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.

Customer collaboration:
Previously there would only be a contract between the customer and the developer. So after the project would be completed it would be handed over to the customer. Many times a situation would arise that the product asked for and the product delivered was not the same. Thus agile insists on continuous delivery so that the developer and the customer are on the same page and can react to changing market conditions.

Responding to change:
According to the Agile Manifesto, there might be many phases in software development when changes can be done to the product. These changes should be encouraged irrespective of the phase the project is in. This helps in achieving better goals and revised results.

2.Write a short notes on the DevOps Orchestration.

- DevOps is a set of practices, principles, and cultural philosophies that emphasize collaboration and communication between software development (Dev) and IT operations (Ops) teams.
- The main objective of DevOps is to enhance the efficiency and reliability of the software development and delivery process by promoting automation, continuous integration, continuous delivery, and other practices that streamline the entire lifecycle.

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.

Reasons to invest in DevOps Orchestration:

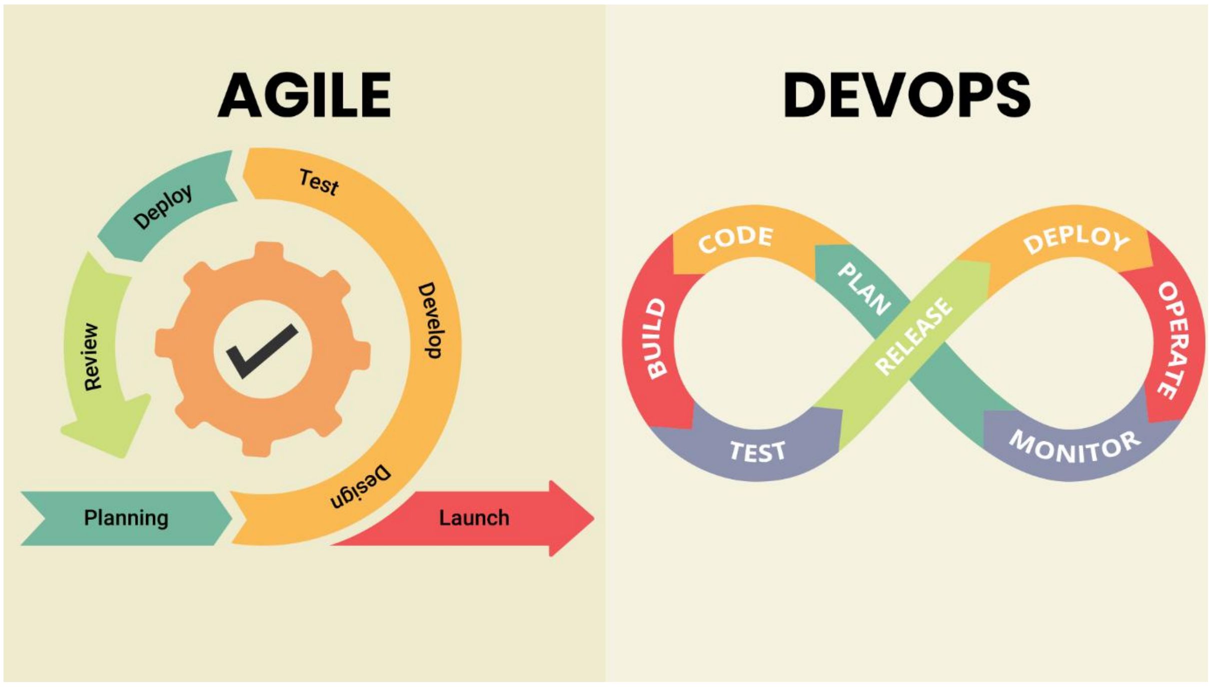
- Speed up the automation process
- Enhance cross-functional collaboration
- Release products with higher quality
- Lower costs for IT infrastructure and human resources
- Build transparency across the SDLC
- Improve the release velocity

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

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.

S. No.	Agile	DevOps
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 Building 2.Sprint advancement	DevOps to have a few best hones that ease the method – 1. Focus 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

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


AGILE
Test
Deploy
Develop
Review
Design
Planning
Launch
DEVOPS
CODE
PLAN
DEPLOY
BUILD
RELEASE
OPERATE
TEST
MONITOR