DevOps

MID-1 QUESTIONS

SET-1

1. Explain importance of Agile software development.

Agile:

Agile methodology is a project management that involves breaking the project into phases and emphasizes continuous collaboration and improvements.

Agile software development is crucial for modern software engineering due to its flexibility, efficiency, and customer-centric approach. Here are some key reasons why Agile is important:

1. Faster Time-to-Market

- Agile follows an iterative approach, allowing teams to deliver small, functional increments quickly.
- Frequent releases ensure that software can reach users faster.

2. Flexibility & Adaptability

- Agile embraces changing requirements, even late in development.
- Teams can adjust priorities based on user feedback or market demands.

3. Enhanced Collaboration

- Agile encourages close collaboration between developers, testers, product managers, and customers.
- Daily stand-up meetings and regular sprint reviews keep everyone aligned.

4. Higher Product Quality

- Continuous testing and integration ensure early detection of bugs.
- Feedback loops help refine features before they reach production.

5. Customer-Centric Approach

- Agile focuses on delivering value to customers through continuous feedback.
- User stories and regular feedback sessions help create software that meets real-world needs.

6. Risk Reduction

- Short development cycles reduce the risk of project failure.
- Regular testing and feedback help identify issues before they become major problems.

7. Improved Team Productivity & Morale

- Agile empowers teams to make decisions and work autonomously.
- A focus on collaboration and continuous improvement fosters motivation.

8. Cost-Effective Development

- Early issue detection reduces expensive fixes in later stages.
- Efficient resource management ensures better cost control.

9. Transparency & Accountability

- Agile methodologies, like Scrum and Kanban, promote visibility into progress and challenges.
- Regular retrospectives help teams improve processes continuously.

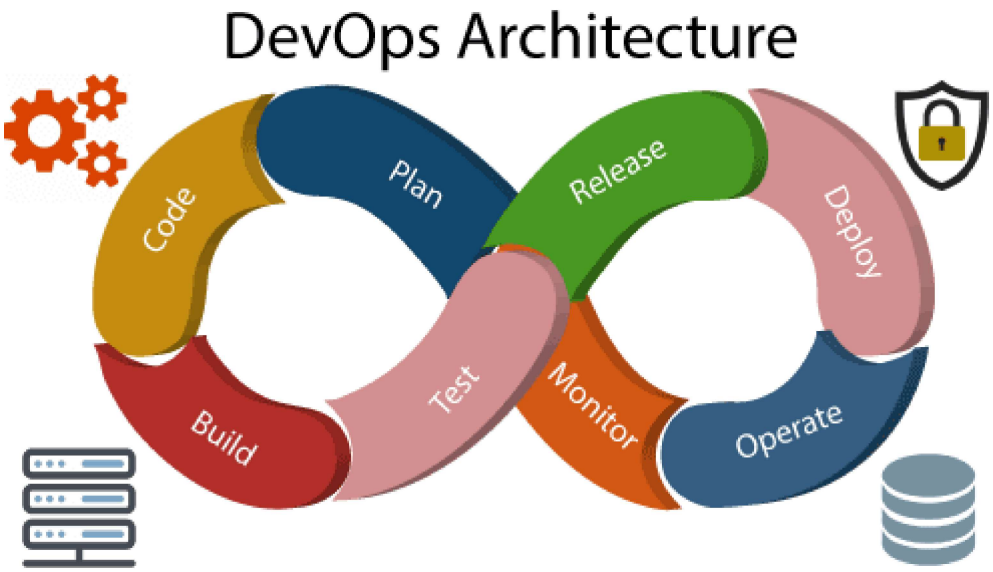
10. Better Stakeholder Engagement

- Regular updates and feedback sessions keep stakeholders informed.
- Involvement in planning and prioritization ensures that business goals are met.

2. Explain DevOps architecture and it’s features with neat sketch.

DevOps :

DevOps is a software development approach that emphasizes collaboration and communication between development and operations teams.



The Features of DevOps :

1) Build

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

2) Code

Many good practices such as Git enables the code to be used, which ensures writing the code for business, helps to track changes, getting notified about the reason behind the difference in the actual and the

expected output, and if necessary reverting to the original code developed. The code can be appropriately arranged in **files, folders**, etc. And they can be reused.

3) Test

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

4) Plan

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

5) Monitor

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

6) Deploy

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

7) Operate

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

with a monitoring plan which serves the IT and business requirements.

8) Release

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

3. Describe various features and capabilities of agile.

Agile :

Agile methodology is a project management that involves breaking the project into phases and emphasizes continuous collaboration and improvements.

Agile is a flexible and iterative approach to software development that focuses on collaboration, customer satisfaction, and continuous improvement. Below are its key **features and capabilities**:

1. Iterative and Incremental Development

- Software is developed in **small increments** rather than a single release.
- Each iteration (or sprint) delivers a working product with new features.

2. Customer-Centric Approach

- Agile emphasizes continuous feedback from **customers and stakeholders**.
- **User stories** help ensure that development aligns with customer needs.

3. Flexibility and Adaptability

- Agile embraces **changing requirements**, even late in the development cycle.
- Teams can quickly adjust priorities based on business or user demands.

4. Continuous Collaboration

- Encourages communication between **developers, testers, designers, and customers**.
- Daily **stand-up meetings** (Scrum) keep everyone aligned and informed.

5. Faster Time-to-Market

- **Frequent releases** ensure quick delivery of features and updates.
- Early market feedback helps improve the product iteratively.

6. Transparency and Visibility

- Agile frameworks (Scrum, Kanban) provide **clear progress tracking**.
- Tools like **burndown charts, task boards, and sprint backlogs** improve visibility.

7. High Product Quality

- **Continuous testing** ensures that defects are detected and fixed early.
- Regular **retrospectives** help teams improve processes and product quality.

8. Risk Reduction

- Frequent testing and iterative releases **minimize project failure risks**.

- Issues are identified and addressed **before they escalate**.

9. Self-Organizing and Cross-Functional Teams

- Teams have **autonomy** to make decisions and improve efficiency.
- Agile teams consist of **developers, testers, designers, and product owners** working together.

10. Automation and Continuous Integration (CI/CD)

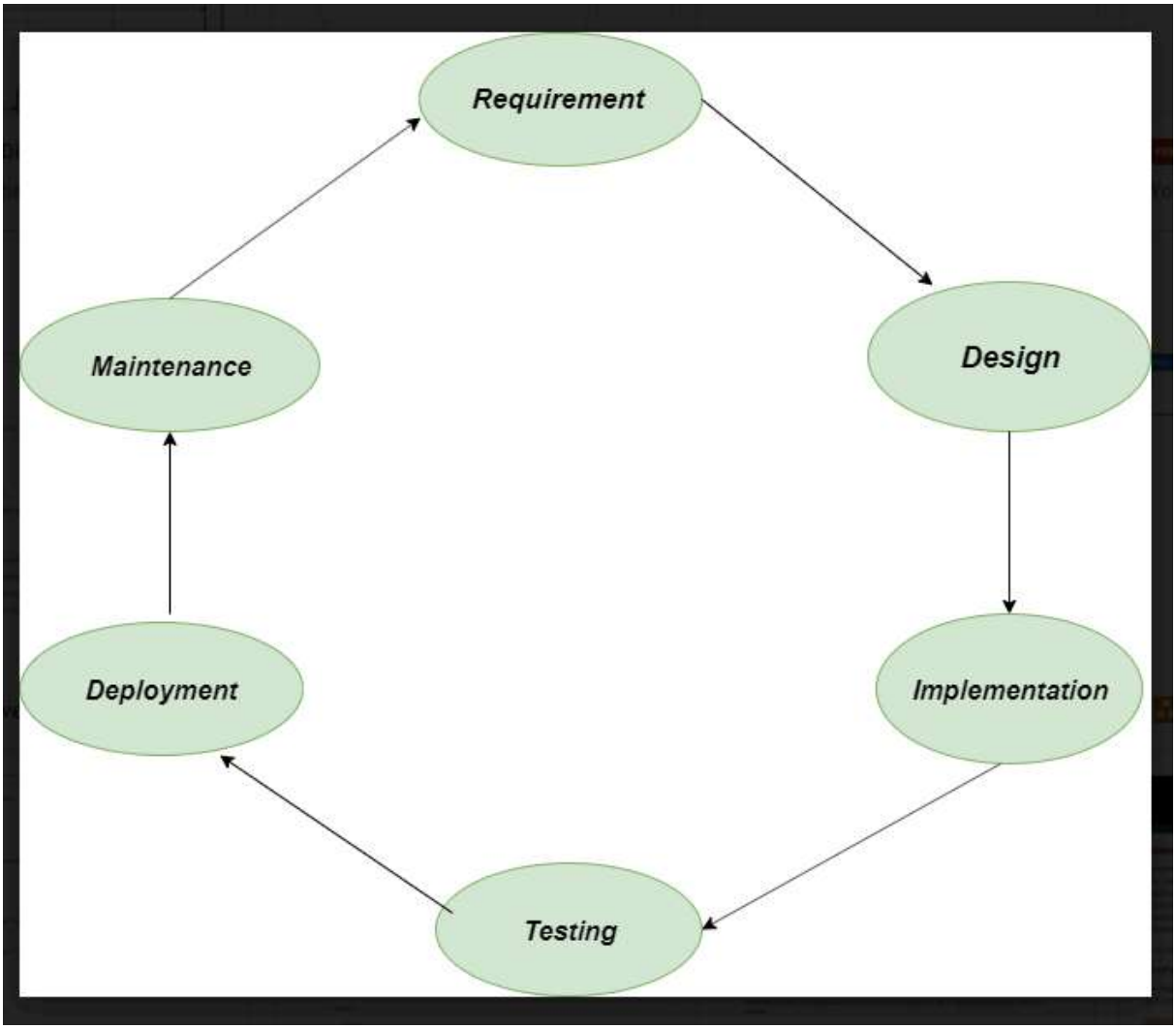
- Encourages **automated testing and deployment** for faster development cycles.
- **Continuous Integration (CI)** helps detect bugs early.

SET-2

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

SDLC(Software Development Life Cycle) :

Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software. SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step. The goal of the SDLC life cycle model is to deliver high-quality, maintainable software that meets the user's requirements. SDLC in software engineering models outlines the plan for each stage so that each stage of the software development model can perform its task efficiently to deliver the software at a low cost within a given time frame that meets users requirements. In this article we will see Software Development Life Cycle (SDLC) in detail.



PHASES IN SDLC :

1.Requirements gathering and analysis :

This phase involves gathering information about the software requirements from stakeholders, such as customers, end-users, and business analysts.

2. Design:

In this phase, the software design is created, which includes the overall architecture of the software, data structures, and interfaces. It has two steps:

High-level design (HLD): It gives the architecture of software products.

Low-level design (LLD): It describes how each and every feature in the product should work and every component.

3. Implementation or coding:

The design is then implemented in code, usually in several iterations, and this phase is also called as Development.

Things you need to know about this phase:

-> This is the longest phase in SDLC model.

-> This phase consists of Front end + Middleware + Back-end.

In front-end: Development of coding is done even SEO settings are done.

In Middleware: They connect both the front end and back end.

In the back-end: A database is created.

4. Testing:

The software is thoroughly tested to ensure that it meets the requirements and works correctly.

5. Deployment:

After successful testing, The software is deployed to a production environment and made available to end-users.

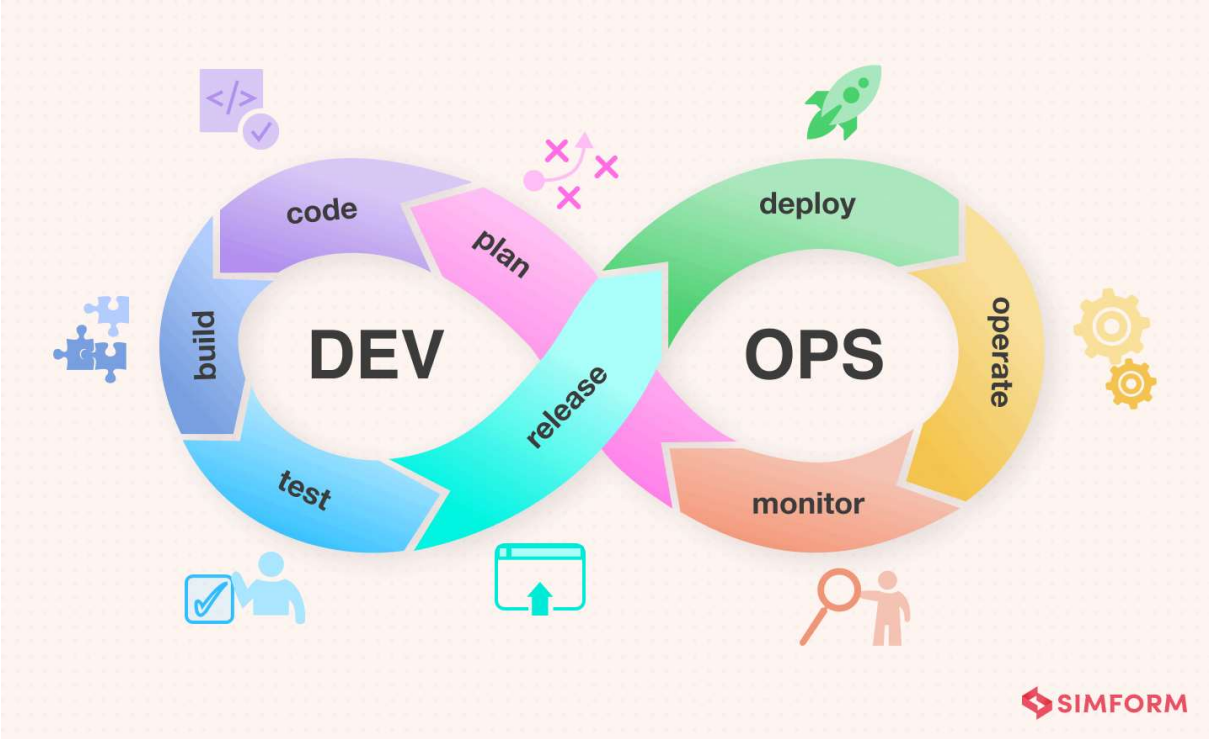
6. Maintenance:

This phase includes ongoing support, bug fixes, and updates to the software.

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

DevOps :

DevOps is a relationship between “Software Development” and “IT Operations” that emphasizes the communication , collaboration and integration.



Stages involved in DevOps Pipeline :

1. Continuous Development
2. Continuous Integration
3. Continuous Testing
4. Continuous Deployment/Continuous Delivery
5. Continuous Monitoring
6. Continuous Feedback
7. Continuous Operations

Continuous Development :

In Continuous Development code is written in small, continuous bits rather than all at once, Continuous Development is important in DevOps because this improves efficiency every time a piece of code is created, it is tested, built, and deployed into production. Continuous Development raises the standard of the code and streamlines the process of repairing flaws, vulnerabilities, and defects. It facilitates developers' ability to concentrate on creating high-quality code.

Continuous Integration :

Continuous Integration can be explained mainly in 4 stages in DevOps. They are as follows:

-> Getting the SourceCode from SCM

-> Building the code

-> Code quality review

-> Storing the build artifacts

The stages mentioned above are the flow of Continuous Integration and we can use any of the tools that suit our requirement in each stage and of the most popular tools are GitHub for source code management(SCM) when the developer develops the code on his local machine he pushes it to the remote repository which is GitHub from here who is having the access can Pull, clone and can make required changes to the code. From there by using Maven we can build them into the required package (war, jar, ear) and can test the Junit cases.SonarQube performs code quality reviews where it will measure the quality of source code and generates a report in the form of HTML or PDF format. Nexus for storing the build artifacts will help us to store the artifacts that are build by using Maven and this whole process is achieved by using a Continuous Integration tool Jenkins.

Continuous Testing :

Any firm can deploy continuous testing with the use of the agile and DevOps methodologies. Depending on our needs, we can perform continuous testing using automation testing tools such as Testsigma, Selenium, LambdaTest, etc. With these tools, we can test our code and prevent problems and code smells, as well as test more quickly and intelligently. With the aid of a continuous integration platform like Jenkins, the entire process can be automated, which is another added benefit.

Continuous Deployment/Continuous Delivery :

Continuous Deployment : Continuous Deployment is the process of automatically deploying an application into the production environment when it has completed testing and the build stages. Here, we'll automate everything from obtaining the application's source code to deploying it.

Continuous Delivery: Continuous Delivery is the process of deploying an application into production servers manually when it has completed testing

and the build stages. Here, we'll automate the continuous integration processes, however, manual involvement is still required for deploying it to the production environment.

Continuous Monitoring :

DevOps lifecycle is incomplete if there was no Continuous Monitoring. Continuous Monitoring can be achieved with the help of Prometheus and Grafana we can continuously monitor and can get notified before anything goes wrong with the help of Prometheus we can gather many performance measures, including CPU and memory utilization, network traffic, application response times, error rates, and others. Grafana makes it possible to visually represent and keep track of data from time series, such as CPU and memory utilization.

Continuous Feedback :

Once the application is released into the market the end users will use the application and they will give us feedback about the performance of the

application and any glitches affecting the user experience after getting multiple feedback from the end users' the DevOps team will analyze the feedbacks given by end users and they will reach out to the developer team tries to rectify the mistakes they are performed in that piece of code by this we can reduce the errors or bugs that which we are currently developing and can produce much more effective results for the end users also we reduce any unnecessary steps to deploy the application. Continuous Feedback can increase the performance of the application and reduce bugs in the code making it smooth for end users to use the application.

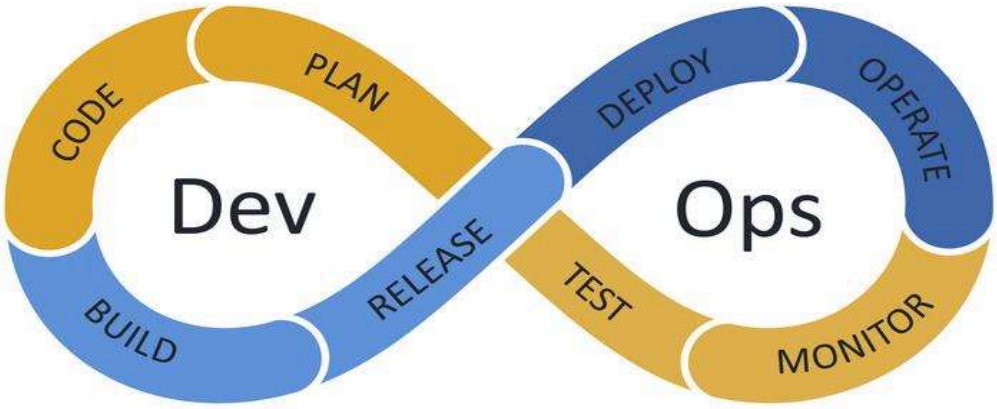
Continuous Operations :

We will sustain the higher application uptime by implementing continuous operation, which will assist us to cut down on the maintenance downtime that will negatively impact end users' experiences. More output, lower manufacturing costs, and better quality control are benefits of continuous operations.

3.Describe the phases in DevOps life cycle.

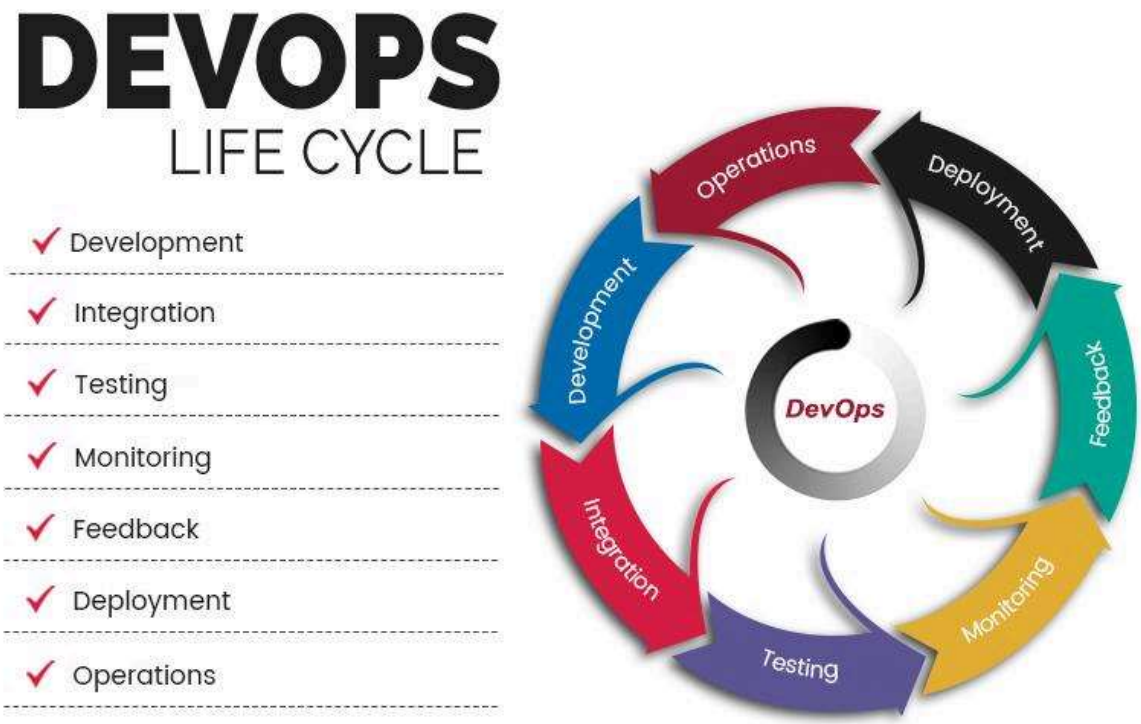
DevOps :

DevOps is a relationship between "Software Development" and "IT Operations" that emphasizes the communication , collaboration and integration.



DevOps Lifecycle :

DevOps is a practice that enables a single team to handle the whole application lifecycle, including development, testing, release, deployment, operation, display, and planning. It is a mix of the terms “Dev” (for development) and “Ops” (for operations). We can speed up the delivery of applications and services by a business with the aid of DevOps. Amazon, Netflix, and other businesses have all effectively embraced DevOps to improve their customer experience.



Stages involved in DevOps Lifecycle :

1. Continuous Development
2. Continuous Integration
3. Continuous Testing
4. Continuous Deployment/Continuous Delivery

5. Continuous Monitoring

6. Continuous Feedback

7. Continuous Operations

Continuous Development :

In Continuous Development code is written in small, continuous bits rather than all at once, Continuous Development is important in DevOps because this improves efficiency every time a piece of code is created, it is tested, built, and deployed into production. Continuous Development raises the standard of the code and streamlines the process of repairing flaws, vulnerabilities, and defects. It facilitates developers' ability to concentrate on creating high-quality code.

Continuous Integration :

Continuous Integration can be explained mainly in 4 stages in DevOps. They are as follows:

-> Getting the SourceCode from SCM

-> Building the code

-> Code quality review

-> Storing the build artifacts

The stages mentioned above are the flow of Continuous Integration and we can use any of the tools that suit our requirement in each stage and of the most popular tools are GitHub for source code management(SCM) when the developer develops the code on his local machine he pushes it to the remote repository which is GitHub from here who is having the access can Pull, clone and can make required changes to the code. From there by using Maven we can build them into the required package (war, jar, ear) and can test the Junit

cases.SonarQube performs code quality reviews where it will measure the quality of source code and generates a report in the form of HTML or PDF format. Nexus for storing the build artifacts will help us to store the artifacts that are build by using Maven and this whole process is achieved by using a Continuous Integration tool Jenkins.

Continuous Testing :

Any firm can deploy continuous testing with the use of the agile and DevOps methodologies. Depending on our needs, we can perform continuous testing using automation testing tools such as Testsigma, Selenium, LambdaTest, etc. With these tools, we can test our code and prevent problems and code smells, as well as test more quickly and intelligently. With the aid of a continuous integration platform like Jenkins, the entire process can be automated, which is another added benefit.

Continuous Deployment/Continuous Delivery :

Continuous Deployment : Continuous Deployment is the process of automatically deploying an application into the production environment when it has completed testing and the build stages. Here, we'll automate everything from obtaining the application's source code to deploying it.

Continuous Delivery: Continuous Delivery is the process of deploying an application into production servers manually when it has completed testing

and the build stages. Here, we'll automate the continuous integration processes, however, manual involvement is still required for deploying it to the production environment.

Continuous Monitoring :

DevOps lifecycle is incomplete if there was no Continuous Monitoring. Continuous Monitoring can be achieved with the help of Prometheus and Grafana we can continuously monitor and can get notified before anything goes wrong with the help of Prometheus we can gather many performance measures, including CPU and memory utilization, network traffic, application

response times, error rates, and others. Grafana makes it possible to visually represent and keep track of data from time series, such as CPU and

memory utilization.

Continuous Feedback :

Once the application is released into the market the end users will use the application and they will give us feedback about the performance of the application and any glitches affecting the user experience after getting multiple feedback from the end users' the DevOps team will analyze the feedbacks given by end users and they will reach out to the developer team tries to rectify the mistakes they are performed in that piece of code by this we can reduce the errors or bugs that which we are currently developing and can produce much more effective results for the end users also we reduce any unnecessary steps to deploy the application. Continuous Feedback can increase the performance of the application and reduce bugs in the code making it smooth for end users to use the application.

Continuous Operations :

We will sustain the higher application uptime by implementing continuous operation, which will assist us to cut down on the maintenance downtime that will negatively impact end users' experiences. More output, lower manufacturing costs, and better quality control are benefits of continuous operations.

SET-3

1.Write the difference between Waterfall and Agile models?

Agile Project Management	Waterfall Project Management
Client input is required throughout the product development.	Client input is required only after completing each phase.

Agile Project Management	Waterfall Project Management
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 Agile project management.	It requires a large team.
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 ecosystem?

DevOps :

DevOps is a relationship between “Software Development” and “IT Operations” that emphasizes the communication , collaboration and integration.

The DevOps ecosystem refers to the collection of tools, practices, and cultural philosophiesthat together enable organizations to implement DevOps principles and practices. DevOps seeks to improve collaboration between development (Dev) and operations (Ops) teams todeliver high-quality software faster and more reliably. The ecosystem encompasses a widerange of tools and technologies that span the entire software development lifecycle (SDLC),from planning and coding to deployment and monitoring.

Key Components of the DevOps Ecosystem:

1. Collaboration and Communication Tools: Effective collaboration and communication between development, operations, and other stakeholders are central to the success of DevOps. Tools in this category help teams work together efficiently, regardless of their location.

2. Continuous Integration (CI) Tools: Continuous Integration (CI) is a practice where developers frequently commit their code changes to a shared repository. CI tools automate the build and testing process, ensuring that new code integrates seamlessly into the codebase.

3. Continuous Delivery (CD) Tools: Continuous Delivery (CD) is an extension of CI, where code changes are automatically deployed to production or staging environments after passing automated tests.It ensures that the latest code is always ready for release.

4. Configuration Management Tools: Configuration management tools automate the process of managing system configurations, ensuring consistency across different environments, including production and development environments. These tools help in maintaining infrastructure as code.

5. Infrastructure Automation and Orchestration Tools: These tools automate the provisioning,management, and scaling of infrastructure, ensuring that it is consistent, repeatable, and scalable. This category includes Infrastructure as Code (IaC) practices that allow infrastructure to be managed in code form.

6. Containerization and Virtualization: Containers and virtualization technologies enable consistent environments across different stages of development, testing, and production. This ensures that applications run the same way, regardless of the environment.

7. Monitoring and Logging Tools: Monitoring and logging are critical in a DevOps environment, as they provide real-time insights into application performance, system health, and user behavior. These tools help teams quickly identify and resolve issues before they impact users.

8. Security Tools (DevSecOps): DevOps incorporates security throughout the software development lifecycle, known as DevSecOps. The aim is to identify security issues early in the development process,making security an integral part of DevOps workflows.

9. Testing Tools: Automated testing is essential for the DevOps pipeline to ensure that code is of high quality and that defects are caught early in the development process.

Benefits of DevOps Ecosystem:

->Improves software delivery speed.

->Enhances collaboration between teams.

->Reduces deployment failures and downtime.

->Ensures continuous monitoring and feedback.

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

To successfully adopt DevOps in IT projects, organizations should follow a series of strategic steps that integrate cultural changes, process improvements, and technological implementations. Here's a breakdown of these steps:

->Cultural Shift.

->Tools and Automation.

->Continuous Testing.

->Continuous Monitoring.

->Feedback Loops.

Cultural Shift: Adopt a DevOps mindset by enabling engineers to cross the barrier between development and operations teams. Encourage open communication and knowledge sharing to improve the efficiency of the software development lifecycle, allowing faster feature delivery and promoting a culture of continuous feedback and enhancement.

Tools and Automation: Identify and plan to automate any manual and repetitive processes. Automate continuous integration, continuous delivery, automated testing,and infrastructure as code. Automating repetitive tasks reduces human error and speeds up the development cycle.

Continuous testing: DevOps emphasizes continuous testing throughout the software development to ensure that code is delivered with high quality and free errors or bugs or defects.

Continuous monitoring: Implement real-time monitoring tools to track application performance, identify potential issues, and receive alerts proactively.

Feed back Loops: DevOps requires the feedback loops to enable continuous improvement. This includes a loop between the development and operations team as well as between the software and users.

SET-4

1.Explain the values and principles of Agile model.

The Agile model is guided by the Agile Manifesto, which articulates four core values and twelve principles that shape Agile practices in software development. These values and principles emphasize flexibility, collaboration, and customer satisfaction, distinguishing Agile from traditional methodologies.

Four Values of the Agile Manifesto :

1. Individuals and Interactions Over Processes and Tools: This value prioritizes human communication and collaboration over rigid processes and tools. It recognizes that successful software development relies on effective teamwork and interpersonal relationships. Agile teams are encouraged to communicate openly, adapt to changes quickly, and respond to customer needs in real-time.

2. Working Software Over Comprehensive Documentation: The focus here is on delivering functional software rather than getting bogged down by extensive documentation. While documentation is still important, the primary goal is to produce a working product that meets user requirements.This approach allows teams to deliver software faster and incorporate feedback more effectively.

3. Customer Collaboration Over Contract Negotiation: Agile emphasizes ongoing collaboration with customers throughout the development process rather than merely negotiating contracts at the project's outset. Engaging customers continuously helps ensure that their needs are met and allows for adjustments based on their feedback, leading to higher satisfaction with the final product.

4. Responding to Change Over Following a Plan: Agile methodologies embrace change, even late in development, recognizing that adapting to new information or shifting requirements can provide a competitive advantage. This flexibility allows teams to pivot when necessary and incorporate new ideas or improvements into the project.

Twelve Principles of the Agile Manifesto :

1. Customer Satisfaction Through Early and Continuous Delivery: Deliver valuable software early and continuously to satisfy customers, ensuring they see progress regularly.

2. Welcome Changing Requirements: Embrace changes in requirements, even late in development, as they can lead to better outcomes for the customer.

3. Deliver Working Software Frequently: Aim for shorter timescales between releases, delivering working software every few weeks or months.

4. Business People and Developers Must Work Together Daily: Foster close collaboration between business stakeholders and developers throughout the project.

5. Build Projects Around Motivated Individuals: Provide a supportive environment for motivated individuals, trusting them to get the job done.

6. Face-to-Face Conversation is the Most Effective Method of Communication: Prioritize direct communication among team members as it enhances understanding and speeds up decision-making.

7. Working Software is the Primary Measure of Progress: Focus on delivering functional software as the main indicator of progress rather than relying solely on documentation or plans.

8. Sustainable Development: Promote a constant pace of work that can be maintained indefinitely without burnout or stress.

9. Technical Excellence and Good Design Enhance Agility: Encourage technical excellence and good design practices to improve agility and adaptability in development efforts.

10. Simplicity is Essential: Maximize the amount of work not done by focusing on simplicity; this helps streamline processes and reduce complexity.

11. Self-Organizing Teams Produce the Best Results: Allow teams to self-organize, fostering creativity and innovation while encouraging ownership of their work.

12. Regularly Reflect on How to Become More Effective: Conduct regular retrospectives to reflect on processes and identify opportunities for improvement continuously.

2. Write a short notes on the DevOps Orchestration.

DevOps orchestration is a critical aspect of the DevOps ecosystem, focusing on the automated coordination and management of various tasks and processes throughout the software development lifecycle. It aims to streamline workflows, enhance efficiency, and reduce the complexities associated with managing multiple automated tasks.

Definition of DevOps Orchestration :

DevOps orchestration refers to the process of automating and coordinating multiple independent automated tasks to create a cohesive workflow within the DevOps pipeline. Unlike automation, which typically focuses on individual tasks, orchestration manages the interactions between these tasks to ensure they work together seamlessly. This includes integrating continuous integration (CI), continuous delivery (CD), deployment processes, and other automated functions like testing and monitoring.

Key Functions of DevOps Orchestration :

1. Workflow Management: Orchestration simplifies complex workflows by managing dependencies and sequences of tasks. This ensures that each step in the software delivery process is executed in the correct order, reducing bottlenecks and potential errors.

2. Automation Coordination: It coordinates various automation tools and processes,allowing them to operate as a unified system. This integration helps in achieving greater efficiency and effectiveness in software development and deployment.

3. Continuous Integration and Delivery: Orchestration plays a vital role in CI/CD practices by automating the integration of code changes, running tests, and deploying applications. This allows for rapid feedback loops and quicker releases to production.

4. Resource Management: By automating resource allocation and configuration management, orchestration helps optimize resource usage across development environments, leading to cost savings and improved performance.

The benefits of DevOps orchestration :

1. Faster resolution of problems

DevOps orchestration helps streamline the problem-solving processes. It accomplishes this by automating responses to alerts and incidents. Thus, you can quickly fix issues and troubleshoot as a team. As a result, you'll have less downtime and minimal disruption of services.

2. Quicker releases of software

DevOps orchestration allows for more rapid software releases. It accomplishes this by eliminating the need for you to wait on other workers to finish their manual duties. This makes it possible for software to reach the end user earlier on, while waiting time can be redirected to the next project. You can accelerate your time to market and expand your service offerings through increased automation, which can reduce costs and boost revenue.

3. Easier management of complexity

Orchestration makes complex systems and multistep processes more manageable. It automates and coordinates tasks across systems and development and operations teams. The likelihood of bottlenecks and miscommunication is reduced. This method is far more efficient than manual management, which often experiences delay.

4. Better resource management

With DevOps orchestration, you can automatically manage resources by scaling them up or down based on real-time demand, ensuring optimal performance. You can also prevent resource overuse by dynamically allocating only what's needed and thus reducing your operational costs.

5. Higher quality of releases

By implementing quality control measures and activities such as security testing approvals, scheduling and status reporting automatically, we can reduce the possibility of errors that affect end-users.

6. Less viable to human error

Automation reduces human errors, which also relates to higher quality of software releases.

7. Seamless collaboration across teams

Orchestration improves the collaboration and togetherness of development and operations teams. Having a platform where all processes and activities are streamlined, consolidated, and updated encourages more collaboration, communication, and team alignment.

8. Helps target key goals and approvals

Teams can automate workflow by routing tasks to the right people for approvals and key decisions. This eliminates delays by ensuring the right person handles each step in the process at the right time. You'll have clear visibility into goals and milestones, and teams can track progress more efficiently.

9. Safeguards and sets up security checks

Automating security checks throughout the development process ensures that vulnerabilities are caught early, reducing the risk of introducing insecure code into production. Orchestration ensures that security testing is conducted at each relevant stage, adding a layer of defence while maintaining development speed.

10. More transparency across the Software Development Life Cycle (SDLC)

When a project's information and tasks are siloed, it becomes hard to get things clear and open. By coordinating all tasks with orchestration in DevOps and making it centralised, we can involve stakeholders better throughout the development lifecycle, and give them more updates and insights into progress.

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.
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	DevOps features a diverse approach and is very viable, most of the time it takes after “Divide and Conquer”. Work partitioned among the

S. No.	Agile	DevOps
	can loan help instead of holding up for the group leads or any pro impedances.	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.
15.	Working system gets more significance in Agile than documentation.	The process documentation is significant in DevOps.