DevOps……

Set-1

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

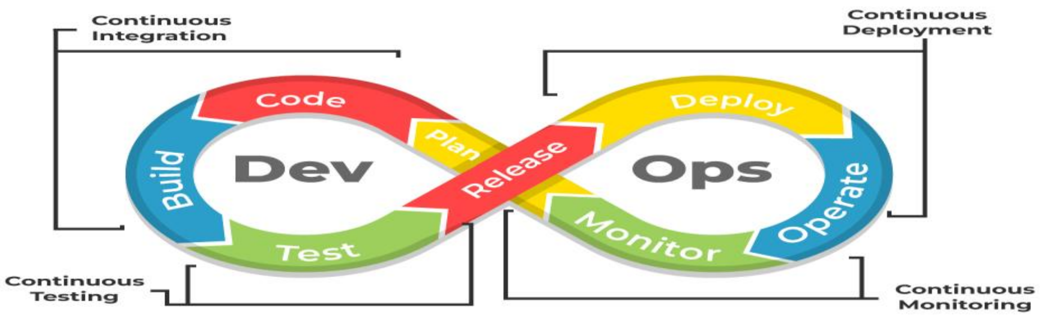
A) Agile software development is crucial for a number of reasons, making it one of the most popular methodologies in the software industry. Here's why it's so important:

1. **Flexibility and Adaptability**: Agile allows teams to respond quickly to changing requirements. As the market or customer needs evolve, Agile teams can adapt their approach and adjust priorities, making it much more flexible compared to traditional methods like Waterfall.
2. **Customer Satisfaction**: Agile emphasizes close collaboration with customers, allowing them to provide feedback regularly throughout the development process. This ensures the final product is closely aligned with customer needs and expectations.
3. **Faster Time-to-Market**: Agile uses iterative cycles (called sprints) to deliver small, functional portions of the software. This approach speeds up the overall development process, enabling teams to release usable features quickly and frequently.
4. **Continuous Improvement**: With Agile, teams regularly review and reflect on their work, encouraging constant improvement in both the product and the process. This iterative approach helps in identifying areas for enhancement early on.
5. **Improved Collaboration and Communication**: Agile promotes daily stand-up meetings, sprint reviews, and retrospectives, encouraging better communication among team members and stakeholders. This leads to quicker problem-solving and stronger teamwork.
6. **Higher Quality**: Since Agile development involves frequent testing and feedback, it reduces the likelihood of bugs and issues piling up. Testing is integrated throughout the process, leading to higher-quality software.
7. **Risk Management**: Because Agile focuses on delivering small chunks of functionality frequently, it helps identify risks early. Teams can address issues as they arise, preventing problems from becoming larger or more costly.
8. **Enhanced Transparency**: Agile makes the development process more visible to all stakeholders. Regular updates, reviews, and demos help everyone stay informed and make it easier to track progress.

In summary, Agile is all about flexibility, customer focus, collaboration, and continuous delivery. It helps teams build better software in a more efficient, responsive, and customer-centered way.

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

A)



DevOps Architecture:

The DevOps architecture typically consists of several components and stages, each contributing to the continuous integration and delivery (CI/CD) pipeline. The architecture often includes the following stages:

1. **Development**: The development team writes code and performs unit testing, version control, and other tasks. The code is stored in a version control system (e.g., Git).
2. **Continuous Integration (CI)**: The code is integrated into a shared repository frequently (multiple times a day). Automated tests are run to detect bugs or integration issues early.
3. **Continuous Testing**: Automated tests, including unit tests, integration tests, and performance tests, are executed during the CI process to ensure the software meets quality standards.
4. **Continuous Deployment (CD)**: After testing, the code is deployed to staging or production environments. This is typically automated to avoid errors during manual deployment and to ensure faster releases.
5. **Monitoring**: Continuous monitoring helps track the performance of the system in real-time, providing visibility into production environments. This step allows teams to detect issues early, like bugs or performance degradation.
6. **Feedback**: The feedback loop helps continuously improve the software development lifecycle based on data from monitoring, tests, and user feedback. Feedback is used for refining features, identifying bugs, and optimizing performance.

Features of DevOps Architecture:

1. **Automation**: Automation pla
   ys a major role in DevOps, from code testing to deployment, ensuring that processes are streamlined and errors are minimized.
2. **Collaboration**: DevOps encourages close collaboration between development and operations teams, breaking down traditional silos to achieve common goals and faster delivery.
3. **Continuous Integration and Delivery (CI/CD)**: Automated processes enable frequent code integration and rapid delivery, making sure that software is constantly being improved and released.
4. **Infrastructure as Code (IaC)**: With IaC, infrastructure is managed and provisioned using code and automation, ensuring consistency and reducing manual errors.
5. **Monitoring and Logging**: Continuous monitoring of applications and infrastructure helps to detect performance issues early, analyze logs, and improve troubleshooting processes.

6. **Scalability and Flexibility**: DevOps supports scalable and flexible infrastructure, allowing businesses to rapidly adjust to changing demands through cloud services and containerization technologies like Docker and Kubernetes.
7. **Security**: DevOps also integrates security practices into the pipeline (DevSecOps), ensuring that security concerns are addressed early in the development lifecycle.

3).Describe various features and capabilities in agile.

A) Agile is a flexible and iterative approach to software development that emphasizes collaboration, customer feedback, and small, incremental improvements. The Agile methodology has several features and capabilities that make it ideal for rapidly changing environments. Let's break them down:

Key Features of Agile:

1. **Iterative and Incremental Development:**
   - **Iterative**: Work is divided into smaller, manageable iterations (called "sprints" in Scrum). Each iteration delivers a functional portion of the product.
   - **Incremental**: New features are added to the product in small, usable increments. This ensures that the product is always progressing and that valuable feedback can be incorporated early.
2. **Collaboration and Communication:**
   - Agile emphasizes **close collaboration** between developers, business stakeholders, and customers. Frequent and transparent communication helps ensure the product aligns with customer expectations.
   - **Daily stand-up meetings** are common where teams quickly sync on progress, roadblocks, and plans.
3. **Customer-Centric Focus:**
   - Agile encourages customer feedback throughout the development process. Regular releases allow the customer to provide input and evaluate progress.
   - **Product Backlog**: Agile teams maintain a backlog of features, and the highest-priority items are worked on first based on customer needs.
4. **Flexibility and Adaptability:**
   - Agile is designed to **respond to changes** in requirements. As customer needs evolve, teams can adapt by updating the backlog and modifying features.
   - Changes can be incorporated into the product at any stage without disrupting the entire workflow.
5. **Simplicity:**
   - Agile stresses **simplicity**—doing just enough work to satisfy the current needs of the project without over-engineering solutions or adding unnecessary features.
6. **Transparency:**
   - Teams, stakeholders, and customers all have visibility into the development process, ensuring **transparency** about progress, challenges, and risks.
   - **Burndown charts** and other visual tools are often used to track progress.

Capabilities of agile

1. Flexibility and Adaptability:

- Agile teams are capable of **responding to changing requirements** even late in the development process. Customer feedback and evolving market conditions are taken into

- : account regularly, allowing the product to adapt to new needs or priorities.

2. Collaborative and Cross-Functional Teams:

- Agile fosters **close collaboration** between developers, testers, business stakeholders, and customers. This ensures that decisions are made with input from all relevant parties and promotes a shared responsibility for success.
- **Cross-functional teams** bring together members with diverse skills, ensuring that all aspects of the product, from development to testing and design, are handled within the team itself.

3. Customer-Centric Development:

- Agile's iterative approach allows for **regular customer feedback**. This ensures that the product is continuously aligned with customer needs and expectations. The Agile team can prioritize the most important features and deliver incremental value in each iteration.

4. Transparency and Visibility:

- Agile emphasizes **transparency** through tools like task boards (e.g., Kanban boards) and regular meetings (e.g., sprint reviews and stand-ups). This allows all stakeholders to have visibility into the progress of the project, making it easier to identify risks, bottlenecks, or areas for improvement.

5. Rapid Time-to-Market:

- Agile focuses on delivering functional software **in short, time-boxed iterations** (called sprints), which often last 1–4 weeks. Each sprint produces a working piece of software, allowing teams to release features or improvements quickly, reducing time-to-market.
- This ability to release in smaller increments enables businesses to stay competitive by delivering features more rapidly than traditional development methods.

6. Risk Mitigation:

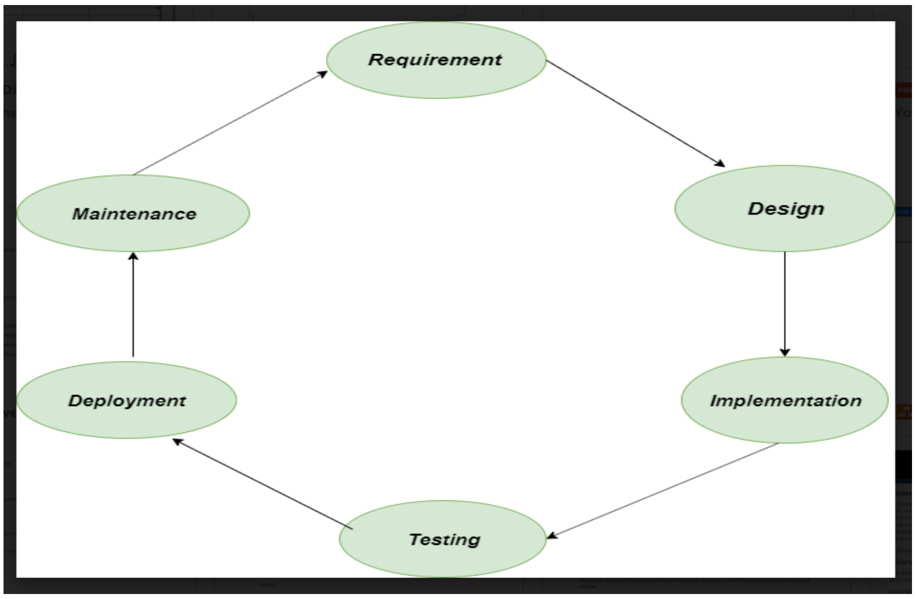
- Agile helps identify risks **early and often**. Regular testing and feedback during sprints make it easier to catch bugs or performance issues early, which reduces the chance of major problems occurring later in development.
- The iterative nature also helps mitigate the risk of building something that the customer doesn't need, as feedback is integrated continuously.

....

Set-2

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

A) The Software Development Life Cycle (SDLC) is a process used by software development organizations to plan, design, develop, test, deploy, and maintain software applications.



The **Software Development Life Cycle (SDLC)** is essential because it offers a structured approach to building software, ensuring all necessary steps are followed for successful development and maintenance. By following SDLC, organizations can:

1. **Ensure Structured Development**: SDLC provides a clear roadmap, ensuring all phases are systematically managed.
2. **Meet Requirements**: It helps align the final product with user needs and business goals.
3. **Minimize Risks**: It identifies and mitigates potential issues early in the process.
4. **Improve Quality**: Continuous testing and feedback ensure higher-quality software.
5. **Support Maintenance**: SDLC ensures software remains adaptable and manageable after deployment.

Phases of SDLC

- **Requirement Gathering and Analysis**:

In this phase, the development team interacts with stakeholders to collect detailed requirements for the software. These requirements are analyzed to ensure they align with business goals and user needs. A requirement specification document is created, which guides the development process.

- **System Design**:

The design phase involves creating the blueprint for the software. It includes defining the system architecture, user interface (UI), and database structure. Design documents like data flow diagrams and wireframes are prepared, which help developers understand how the system will function and interact.

- **Implementation (Coding/Development)**:

This is where the actual development begins. Developers write the code based on the design documents. The software features are developed, and initial functionality is implemented. The system is typically broken down into modules for easier management.

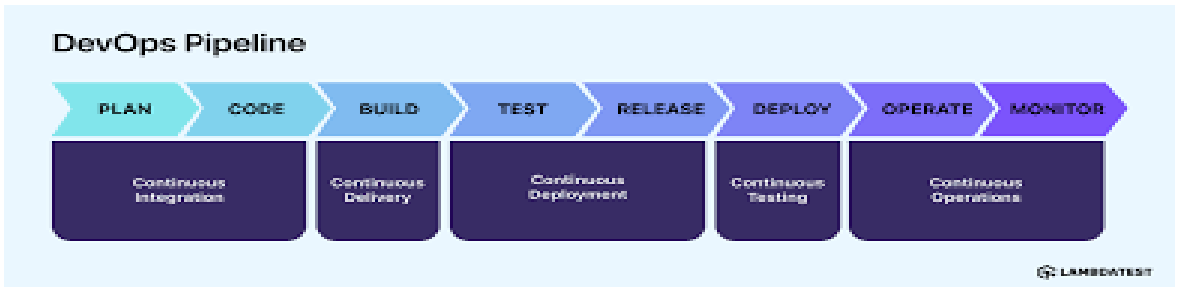
- **Testing**:
After the coding phase, the software goes through various levels of testing (e.g., unit testing, integration testing, system testing) to ensure it functions as expected. Bugs and issues are identified and fixed. Testing ensures the system meets the specified requirements and is free of critical defects.

- **Deployment**:
Once testing is complete and the software is stable, it's deployed to the production environment. This phase includes releasing the software to users, making it available for use, and ensuring that installation and configuration processes go smoothly.

- **Maintenance**:
After deployment, the software enters the maintenance phase. Here, developers address bugs that arise in real-world use, implement updates, and add new features. Regular updates are provided to ensure the software continues to perform well and remains relevant over time.

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

A) The **DevOps pipeline** is a set of stages that automate the process of building, testing, deploying, and monitoring software in a continuous integration and continuous delivery (CI/CD) environment. Here's a brief explanation of the various stages involved:



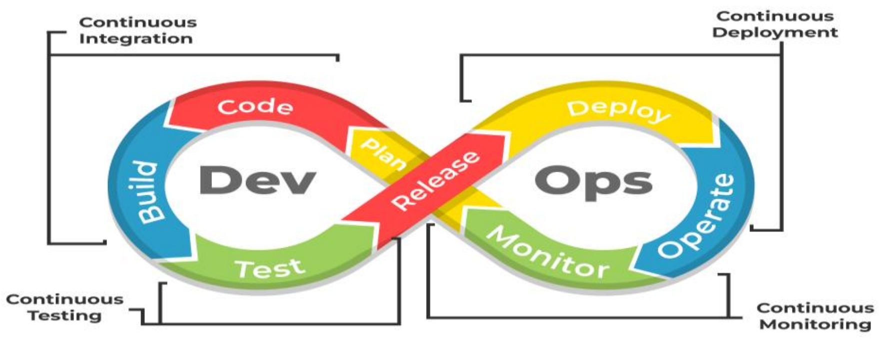
1. **Development**:
Developers write and commit code in this stage. The code is usually stored in a version control system (e.g., Git), where it can be accessed by the whole team for collaboration.
2. **Build**:
In this stage, the code is compiled, and dependencies are installed. Automation tools like Jenkins or GitLab CI are used to create an artifact (such as a compiled application or package) from the source code.
3. **Test**:
The software is automatically tested for bugs and performance issues. Unit tests, integration tests, and functional tests are run to ensure the code works as expected. This helps catch errors early.
4. **Deploy**:
The code is deployed to staging or production environments. This can be an automated deployment to a cloud server or an on-premises system. Continuous deployment ensures that new code is pushed to production without manual intervention.

5. **Operate**:
   Once deployed, the application is monitored in real-time. This stage involves tracking system performance, availability, and user interactions to ensure everything is running smoothly.
6. **Monitor and Feedback**:
   Continuous monitoring tools collect feedback from the operating system, user experience, and error logs. This feedback is used to improve the system and can trigger further iterations in the development process.

The DevOps pipeline helps automate workflows, improve collaboration between teams, and accelerate the delivery of high-quality software.

3). Describe the phases in DevOps life cycle.

A). The **DevOps lifecycle** consists of several phases that aim to streamline and automate the development, deployment, and operation of software applications. Each phase focuses on collaboration between development and operations teams to deliver high-quality software more efficiently and quickly. Here's a breakdown of the key phases:



1. Planning

- **Objective**: Define the goals, features, and requirements for the software.
- In this phase, teams collaborate to understand the product vision and plan features and functionality. This involves deciding the scope, setting deadlines, and aligning on project goals. Tools like Jira or Trello help manage tasks and track progress.

2. Development

- **Objective**: Write and build the application code.
- Developers create the application code and ensure it aligns with the project requirements. During this phase, version control systems like Git are used to manage and track code changes. Agile practices are often employed to break down tasks into smaller, manageable chunks.

3. Build

- **Objective**: Automate the process of compiling and assembling the code.
- This phase focuses on automatically compiling the code and packaging it into a deployable unit. Continuous Integration (CI) tools, such as Jenkins, are used to run automated build

processes. This ensures that the latest code changes are integrated and any potential issues are identified early.

4. Testing

- **Objective**: Ensure the quality of the software by testing it in various scenarios.
- Automated and manual tests are run during this phase to detect bugs, performance issues, and security vulnerabilities. It includes unit testing, integration testing, regression testing, and other quality assurance practices. Continuous testing ensures that software is thoroughly evaluated before moving to production.

5. Release

- **Objective**: Deploy the software to staging or production environments.
- After successful testing, the application is released to staging or directly to the production environment. Continuous Delivery (CD) tools like Kubernetes or Docker automate the deployment process, ensuring that software is quickly and reliably released with minimal manual intervention.

6. Deploy

- **Objective**: Make the application live and accessible to end users.
- The software is deployed into production, where it is accessible to users. This phase may involve load balancing, scaling the application, and ensuring that it is performing well under real-world conditions. Tools like Ansible or Chef may be used to automate the deployment.

7. Operate

- **Objective**: Monitor and maintain the application in production.
- Once the software is deployed, it's actively monitored to ensure its performance, availability, and security. Monitoring tools like Prometheus or Splunk help track system health, application performance, and user interactions, allowing teams to quickly identify and resolve issues.

8. Monitor and Feedback

- **Objective**: Gather feedback to improve the application and process.
- Continuous monitoring provides real-time data about the system's performance and any potential issues. Feedback loops from customers, stakeholders, and monitoring tools are collected and analyzed. This feedback is used to make improvements to the software and the development process, starting the cycle again.

Summary:
The DevOps lifecycle integrates development and operations teams to streamline the entire software delivery process. The stages—from planning to monitoring—ensure that the application is efficiently developed, tested, deployed, and maintained, with continuous feedback enabling constant improvement. This collaborative, automated approach helps reduce manual effort, improve product quality, and accelerate time-to-market.

Set-3

1). Write the difference between Waterfall and Agile models.

A).

Agile Project Management	Waterfall Project Management
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 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.

s

Agile Project Management	Waterfall Project Management
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.

A). DevOps Eco System: DevOps is a cultural and technical movement that aims to improve collaboration and integration between software development (Dev) and IT operations (Ops) teams. The goal is to increase the speed and quality of software development and delivery through automation, continuous integration, and improved collaboration. The DevOps ecosystem is made up of various principles, practices, tools, and stages that support the DevOps lifecycle.

Core Principles of DevOps

1. **Collaboration and Communication**: DevOps encourages the collaboration between developers and operations teams, promoting transparency and shared goals.
2. **Automation**: Automating tasks like code deployment, testing, and infrastructure provisioning ensures consistency and faster releases.
3. **Continuous Integration/Continuous Delivery (CI/CD)**: CI enables frequent integration of code changes, and CD automates the deployment process, leading to faster releases.
4. **Monitoring and Feedback**: Real-time monitoring provides feedback that allows teams to quickly detect issues in production and continuously improve systems.
5. **Infrastructure as Code (IaC)**: Automating infrastructure management with code ensures consistency and easier scaling.

Stages of the DevOps Lifecycle

1. **Plan**: Defines project requirements, user stories, and features.
   - Tools: Jira, Trello, Azure DevOps
2. **Develop**: Developers write code, and Agile methodologies are often used to iterate rapidly.
   - Tools: Git, GitHub, Visual Studio Code
3. **Build**: The code is compiled and packaged for deployment, often using automation.
   - Tools: Jenkins, GitLab CI/CD, CircleCI
4. **Test**: Continuous testing ensures code quality before deployment.
   - Tools: Selenium, JUnit, Postman
5. **Release**: Code is released to staging or production environments automatically.
   - Tools: Jenkins, Octopus Deploy, Spinnaker
6. **Deploy**: Deployment automation is used to push code into production environments.
   - Tools: Kubernetes, Docker, Terraform, AWS

7. **Operate**: Managing the deployed software for uptime and reliability.
   - Tools: Prometheus, Grafana, Nagios
8. **Monitor**: Monitoring performance and collecting metrics to ensure system health.
   - Tools: ELK Stack, Datadog, New Relic

DevOps Tools and Their Roles

- **Version Control**: Git, SVN – Tracks and manages code changes.
- **Build Automation**: Jenkins, Travis CI – Automates the process of compiling and testing code.
- **Containerization & Orchestration**: Docker, Kubernetes – Helps in packaging applications into containers and orchestrating them at scale.
- **Configuration Management**: Ansible, Puppet – Manages system configurations and deployment tasks.
- **Monitoring**: Datadog, Nagios, ELK Stack – Monitors system health, logs, and application performance…..

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

A). Adopting **DevOps** in IT projects involves a series of steps to improve collaboration, automate processes, and deliver high-quality software faster. Below are the key steps involved

1. Assess Current Processes and Identify Gaps

- **Objective**: Analyze existing workflows and identify inefficiencies or bottlenecks.
- **Explanation**: Understand the current state of software development, deployment, and operations to pinpoint areas where DevOps practices can provide improvements.
- **Action**: Review existing processes, tools, and communication methods.

2. Define Clear DevOps Goals and Objectives

- **Objective**: Set measurable goals that align with business needs.
- **Explanation**: DevOps adoption should be goal-driven. Set objectives like reducing deployment time, improving release frequency, and enhancing system reliability.
- **Action**: Establish KPIs such as **deployment frequency** and **time to market**.

3. Build Cross-Functional Teams

- **Objective**: Create teams combining development, operations, and security expertise.
- **Explanation**: DevOps focuses on collaboration across traditional silos. Building cross-functional teams ensures all aspects of the software lifecycle are addressed collaboratively.
- **Action**: Form teams that include developers, operations staff, and QA professionals

4. **Select the Right Tools for Automation and Integration**

- **Objective**: Implement tools that enable automation and streamline workflows.
- **Explanation**: Choose tools that support continuous integration, deployment, and infrastructure management.

s

- **Action**: Implement tools like **Git**, **Jenkins**, **Docker**, and **Kubernetes** for CI/CD, containerization, and orchestration.

5. Implement Continuous Integration and Continuous Deployment (CI/CD)

- **Objective**: Automate integration and deployment processes.
- **Explanation**: Automate the build, test, and deployment pipelines to ensure faster, reliable releases.
- **Action**: Set up automated pipelines using **Jenkins** or **GitLab CI** for code integration and deployment.

6. Foster a Culture of Collaboration

- **Objective**: Promote communication between development, operations, and security teams.
- **Explanation**: Encourage a culture of shared responsibility and continuous feedback for better collaboration and faster issue resolution.
- **Action**: Create shared ownership for both the development and production environments.

7. Automate Testing and Quality Assurance

- **Objective**: Ensure continuous testing of code to maintain quality.
- **Explanation**: Integrate automated testing within the CI/CD pipeline to catch issues early in the development process.
- **Action**: Use tools like **Selenium**, **JUnit**, or **SonarQube** to automate testing.

8. **Monitor and Optimize Continuously**

- **Objective**: Continuously monitor the application's performance and optimize as needed.
- **Explanation**: Real-time monitoring and analysis help identify potential issues before they affect end-users and provide insights for improvement.
- **Action**: Implement monitoring tools like **Prometheus**, **Grafana**, and **Datadog** to track system health and performance.

9. Iterate and Improve

- **Objective**: Continuously refine and optimize processes and tools based on feedback.
- **Explanation**: DevOps adoption is iterative. Review performance, gather feedback, and make improvements to enhance efficiency.
- **Action**: Regularly analyze data and make adjustments to processes for continuous improvement.

Set-4

s

1). Explain the values and principles of Agile model.

A). The **Agile model** is a widely adopted approach in software development that emphasizes flexibility, collaboration, and rapid delivery of working software. It promotes iterative development, where feedback is used to adapt the product and process continually. The **Agile Manifesto** outlines its core values and principles, which guide Agile practices.

key values:

1. **Individuals and Interactions Over Processes and Tools**
   - Face-to-face communication is preferred over lengthy documentation or reliance on specific software.
2. **Working Software Over Comprehensive Documentation**
   - Instead of spending months creating detailed specs, Agile teams focus on delivering a minimum viable product (MVP) early on, with features added iteratively.
3. **Customer Collaboration Over Contract Negotiation**
   - Regular customer feedback and updates are incorporated into the product to ensure it meets evolving requirements.
4. **Responding to Change Over Following a Plan**
   - If market conditions change or a customer requests a new feature, the team can adjust the product backlog and re-prioritize work.

Agile Principles

1. **Customer satisfaction through early and continuous delivery**
   - Deliver valuable software early and frequently.
2. **Welcome changing requirements**
   - Embrace changes, even late in development, to improve the product.
3. **Frequent delivery of working software**
   - Release software often with a preference for short timescales.
4. **Business and developers must work together daily**
   - Close collaboration between stakeholders and developers ensures alignment.
5. **Build projects around motivated individuals**
   - Trust and support teams to get the job done.
6. **Face-to-face communication is most effective**
   - Prefer direct, in-person communication for clarity and efficiency.
7. **Working software is the primary measure of progress**
   - Focus on delivering functional software over other project metrics.
8. **Sustainable development**
   - Maintain a steady, constant pace to avoid burnout and ensure long-term productivity.
9. **Technical excellence and good design enhance agility**
   - Prioritize quality and clean design to improve adaptability.
10. **Simplicity is essential**

- Focus on the essential work that adds value, avoiding unnecessary complexity.

11. **Self-organizing teams produce the best results**

- Empower teams to make decisions and solve problems.

12. **Regular reflection and adaptation**

- Teams should regularly assess their performance and improve their processes.

2). Write a short notes on the DevOps Orchestration.

DevOps Orchestration refers to the coordination and automation of tasks across various tools and processes in the software development pipeline. Its purpose is to streamline and integrate tasks like coding, testing, deployment, and monitoring, ensuring that everything works seamlessly and efficiently.

Key Aspects of DevOps Orchestration:

1. **Automation**: Automates tasks like integration, testing, and deployment to reduce manual intervention and errors.
2. **Tool Integration**: Connects tools (e.g., Git, Jenkins, Kubernetes) to create a smooth workflow from development to production.
3. **CI/CD Pipeline**: Orchestrates continuous integration and deployment for faster, frequent software releases.
4. **Infrastructure as Code (IaC)**: Automates infrastructure management, ensuring consistency across environments.

Benefits:

- **Faster Delivery**: Speeds up the software development process by automating manual tasks.
- **Consistency**: Ensures the same process is followed in different environments.
- **Reduced Errors**: Automation minimizes the risk of human error.
- **Scalability**: Easily handles growing workloads and teams.

Popular Tools:

- **Jenkins**: Automates CI/CD pipelines.
- **Kubernetes**: Orchestrates containerized applications.
- **Ansible**: Manages infrastructure as code.

Conclusion:

DevOps orchestration is crucial for automating workflows, improving collaboration, and delivering software more quickly. By integrating tools and automating processes, it simplifies development and enhances efficiency.

s

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

A).

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	DevOps features a diverse

s

S. No	Agile	DevOps
	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.	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.	A big team for your project is not required.	It demands collaboration among different teams for the completion of work.
11.	**Some of the Tools-** • Bugzilla • JIRA • Kanboard and more.	**Some of the Tools-** • Puppet • Ansible • AWS • Chef • team City OpenStack and more.