Set 1

1. Explain importance of Agile software development.

Agile Software Development is a software development methodology that values flexibility, collaboration, and customer satisfaction. It is based on the Agile Manifesto, a set of principles for software development that prioritize individuals and interactions, working software, customer collaboration, and responding to change

- **Flexibility and Adaptability:** Agile allows teams to adapt quickly to changes in requirements, technology, or customer needs. This is especially useful in industries where customer preferences or business goals can evolve rapidly.

- **Faster Delivery of Working Software:** Agile promotes delivering working software in shorter, manageable increments, typically in 2-4 week cycles known as "sprints." This allows for faster feedback and quicker releases, reducing the time it takes to bring a product to market.

- **Continuous Improvement:** Through regular iterations and retrospective meetings, Agile teams are constantly reflecting on their processes, identifying areas for improvement, and implementing changes in the next sprint. This culture of continuous improvement leads to more efficient and effective development practices over time.

- **Customer-Centric Focus:** Agile emphasizes close collaboration with stakeholders and end users, ensuring that the software being developed aligns closely with customer needs and expectations. Regular feedback from the customer helps guide the direction of the project.

- **Enhanced Collaboration:** Agile encourages frequent communication between developers, testers, and business stakeholders. This collaborative environment ensures that all team members are aligned on goals, challenges, and changes, leading to a more cohesive and focused development effort.

- **Higher Quality:** Agile incorporates testing and quality assurance throughout the development cycle, rather than at the end of the process. Continuous testing helps catch bugs and issues early, ensuring higher-quality software with fewer defects.

- **Reduced Risk:** Since Agile delivers smaller increments of functionality and allows for continuous feedback, the risk of developing features that are not needed or don't meet expectations is minimized. Problems are identified early, making it easier to pivot or adjust.

- **Empowered Teams:** Agile methodologies encourage self-organizing and cross-functional teams, where developers take ownership of their work. This leads to greater creativity, innovation, and accountability, as team members have more autonomy in how they approach their tasks.

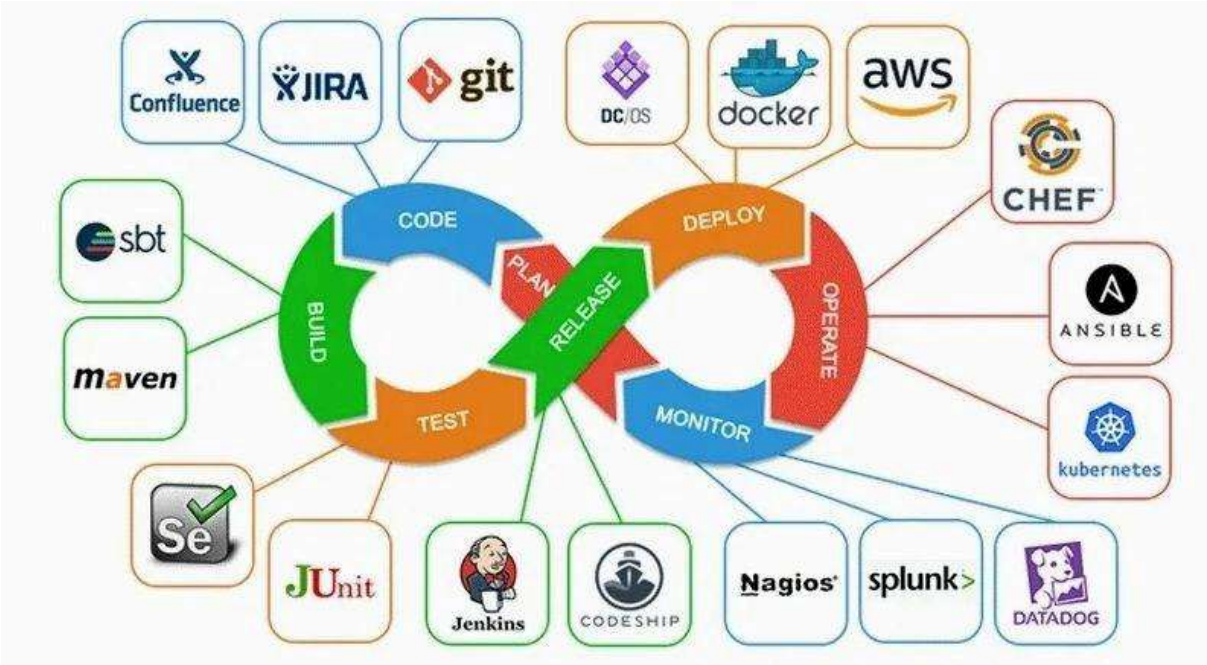
- **Increased Transparency:** With regular standups, sprint reviews, and progress updates, Agile provides transparency into the development process. Both the team and the client can see where the project stands, what has been completed, and what still needs attention.

 Better Risk Management: By focusing on small, manageable tasks and continuous review, Agile helps identify and address risks early. This leads to a more predictable development timeline and fewer unexpected roadblocks.

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

A. DevOps Architecture:

DevOps (Development and Operations) is a set of practices that aims to shorten the system development life cycle (SDLC) and provide continuous delivery with high software quality. It promotes collaboration between development and operations teams to build, test, and release software more efficiently.



Key Components of DevOps Architecture:

DevOps architecture generally involves several components that work together to automate and streamline the software development and deployment process.

1. **Source Code Repository:**
    - This is where all the source code, including the version control system (VCS), is stored. It could be tools like **Git**, **Subversion (SVN)**, or **Mercurial**.
    - Developers commit their code, and version control ensures that code changes are tracked over time.
2. **Continuous Integration (CI):**
    - CI tools like **Jenkins**, **Travis CI**, or **CircleCI** automate the process of building and testing code every time a developer commits changes.
    - Code is continuously integrated into the main codebase to avoid integration issues later on.
3. **Continuous Testing (CT):**
    - Automated testing tools like **Selenium** and **JUnit** run tests on the codebase to detect bugs or issues early in the process.
    - Testing ensures the quality of the software, identifying problems in individual components or integrations.
4. **Continuous Deployment (CD):**
    - **Continuous Deployment** automates the process of deploying code into production environments. Tools like **Ansible**, **Puppet**, or **Chef** manage infrastructure configuration and deployment.
    - This reduces the manual intervention in deployment and speeds up the release cycle.
5. **Infrastructure as Code (IaC):**
    - **Terraform**, **CloudFormation**, or **Ansible** allow teams to define and manage infrastructure (like servers, databases, and networking) through code.
    - This enables faster provisioning and consistent environments across development, testing, and production.
6. **Monitoring and Logging:**
    - Monitoring tools like **Prometheus**, **Nagios**, or **Datadog** track the performance of applications and infrastructure in real-time.
    - Logging tools like **ELK Stack** (Elasticsearch, Logstash, and Kibana) provide detailed logs that help teams identify issues and respond to them quickly.
7. **Collaboration and Communication:**
    - Tools like **Slack**, **Microsoft Teams**, or **JIRA** are used for communication between the development and operations teams.
    - Collaboration ensures that feedback from both teams is quickly addressed.
8. **Containerization and Virtualization:**
    - **Docker** and **Kubernetes** help in packaging applications and their dependencies into containers, which ensures that the software runs consistently across various environments.
    - **Kubernetes** provides orchestration for scaling, deploying, and managing containerized applications.

Features of DevOps Architecture:

1. **Automation:**
    - DevOps automates repetitive tasks like testing, deployment, and configuration management. This ensures faster release cycles and fewer errors.
2. **Collaboration:**

- DevOps emphasizes collaboration between development and operations teams, breaking down traditional silos. This leads to better alignment, communication, and faster decision-making.

3. **Continuous Integration and Continuous Deployment (CI/CD):**
   - The foundation of DevOps is CI/CD pipelines, which automate the process of integrating and deploying code frequently and reliably.
4. **Scalability:**
   - DevOps enables scaling applications and infrastructure easily, especially when using containers and cloud platforms.
5. **Monitoring and Feedback:**
   - Continuous monitoring and feedback help ensure that the application runs smoothly in production, and problems are quickly detected and fixed.
6. **Infrastructure as Code (IaC):**
   - Infrastructure is defined and managed using code, which allows teams to provision environments rapidly and consistently.
7. **Resilience and Reliability:**
   - By using automated tests, monitoring, and rapid feedback loops, DevOps enhances the reliability and resilience of systems, reducing downtime.

3. Describe various features and capabilities in agile.

A. In Agile methodology, "features" represent distinct functionalities or services that deliver business value to users, while "capabilities" are higher-level abilities encompassing multiple features, often spanning across different project components, providing a broader solution set; both are broken down into smaller, manageable user stories for iterative development and delivery, allowing for flexibility and adaptation to changing requirements throughout the project lifecycle.

Key features of Agile:

User-centric focus:

Features are designed to directly address user needs and pain points, prioritizing user experience through clear descriptions of interactions and functionalities.

Independent units:

Features are structured as self-contained components, enabling teams to work on and deliver them incrementally without significant dependencies.

Prioritization:

Features are ranked based on business value and importance, guiding the development team to focus on high-priority items first.

Acceptance criteria:

Each feature has clear and measurable acceptance criteria to define when it is considered complete and ready for delivery.

Iterative development:

Features are delivered in small increments throughout the project, allowing for continuous feedback and adaptation to changing requirements.

Flexibility:

Agile methodology allows for adjustments to features based on new information or changing priorities during development.

Key capabilities in Agile:

Cross-functional collaboration:

Capabilities often involve multiple teams working together to achieve a larger solution, requiring strong communication and coordination across different disciplines.

System-level thinking:

Capabilities represent broader system behaviors or functionalities that might encompass multiple features across different components of a product.

Scalability:

Capabilities can be broken down into smaller features that can be delivered in different project phases or iterations, allowing for scaling based on project complexity.

Strategic alignment:

Capabilities are often used to align features with overall business objectives and strategic goals, ensuring development efforts contribute to the wider vision.

Example of Feature vs Capability:

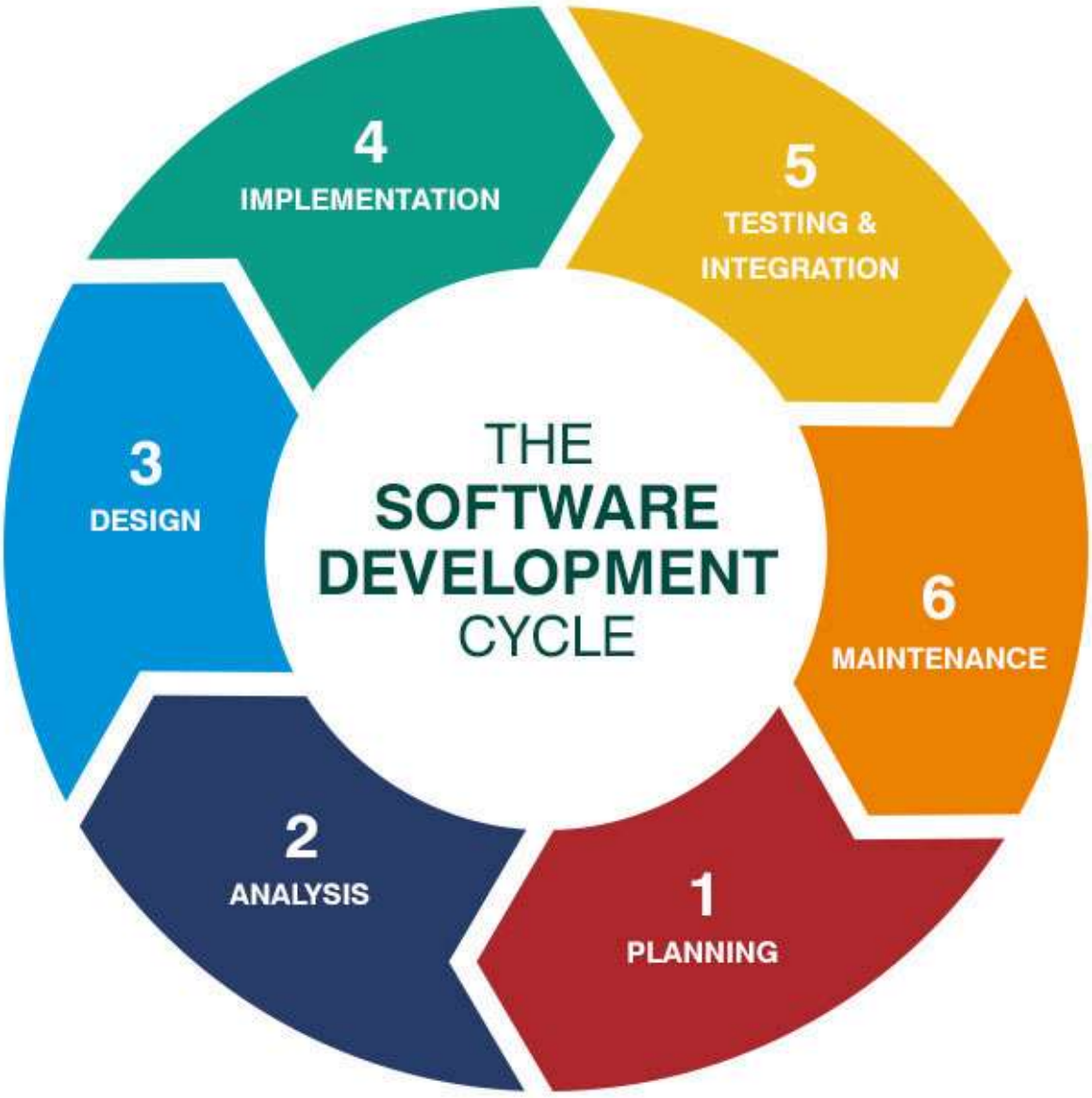
Feature: "Implement a user login functionality with email and password authentication."

Capability: "Enable secure user access to the application across multiple platforms."

Set 2

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

A. **SDLC (Software Development Life Cycle)** is a structured approach or a series of phases that are followed during the development of a software product. It defines the process used by software developers and project managers to design, develop, and maintain high-quality software. SDLC provides a systematic way to plan, create, test, and deliver software applications.



1. Planning and Requirement Analysis:

- **Objective**: The goal of this phase is to gather business requirements and understand the scope of the software project.

2. System Design:

- **Objective**: Create a blueprint of the software system by designing its architecture, components, and interfaces.

3. Implementation (Coding/Development):

- **Objective**: This phase involves the actual development of the software based on the designs.

4. Integration and Testing:

- **Objective**: The goal is to ensure that the software is free of defects and meets all the requirements.

5. Deployment:

- **Objective**: This phase involves releasing the software to a live or production environment.

6. Maintenance:

- **Objective**: Once deployed, the software enters the maintenance phase where ongoing support and updates are handled.

7. Feedback and Review:

- **Objective**: Collect feedback from users and stakeholders to evaluate the software's effectiveness.

8. Retirement or Decommissioning:

- **Objective**: The final phase involves phasing out the software once it is no longer needed or has become obsolete.

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

Helps catch bugs early and reduces the risk of issues in production systems

Continuous testing

Involves automating tests to ensure that new code changes don't negatively impact the application's functionality

Uses automation testing tools like Selenium, TestNG, and JUnit

Continuous deployment

Involves automatically deploying code changes to a production environment

Ensures accurate and smooth deployment without impacting the software's performance

Allows organizations to quickly and frequently deliver new updates and features

Continuous monitoring

Involves observing, measuring, and analyzing the performance and behavior of software applications, servers, and infrastructure components

Helps IT teams quickly identify issues related to app performance

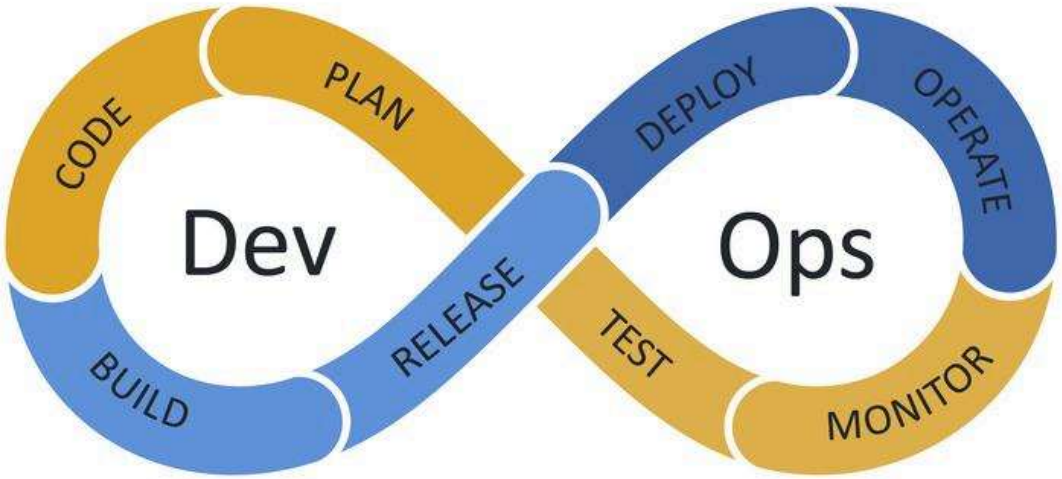
Continuous feedback

Involves continuously monitoring applications and infrastructure to gather insights on performance and user experience

Helps teams quickly identify and address issues

2. Describe the phases in DevOps life cycle.

DevOps is a practice that enables a single team to handle the whole application lifecycle, including development, testing, release, deployment, operation, display, and planning.



1. Planning

- **Objective:** Establish requirements, define project scope, and plan the roadmap.
- **Activities:**
  - Collaborating between development and operations teams to set clear goals and priorities.
  - Identifying features, user stories, and product backlogs.
  - Setting timelines, milestones, and planning releases.

2. Development

- **Objective:** Writing the application code and building the software.
- **Activities:**
  - Developers write the source code for the application.
  - Code is often developed in small, manageable increments.
  - The development environment is managed to maintain consistency between different developers' machines.

3. Building

- **Objective:** Compile code, automate builds, and ensure code quality.
- **Activities:**

    - Continuous Integration (CI) tools are used to automatically compile and build the code.
    - Code is integrated and merged into a shared repository.
    - Build automation tools like Jenkins or Bamboo are used to check code integrity and avoid integration issues.

4. Testing

- **Objective:** Ensure the application is free of bugs and meets functional and non-functional requirements.
- **Activities:**
    - Automated testing is used to verify functionality, performance, and security.
    - Testing includes unit tests, integration tests, regression tests, and performance tests.
    - Continuous Testing ensures the application is constantly being validated at every stage.

5. Release

- **Objective:** Deploy the code to production-ready environments.
- **Activities:**
    - Code is prepared for release and sent to the staging environment.
    - Deployment pipelines are created to automate the release process (using tools like Kubernetes, Docker, etc.).
    - Versioning, configuration management, and monitoring of the release are planned.

6. Deployment

- **Objective:** Deploy the software to the live production environment.
- **Activities:**
    - Continuous Delivery/Deployment ensures that the code is ready to be pushed to production at any time.
    - Infrastructure as Code (IaC) tools like Terraform or Ansible are used to automate and manage infrastructure.
    - Rolling deployments, canary releases, and blue-green deployments are used to reduce downtime and risks.

7. Monitoring

- **Objective:** Track the application's performance and user interactions to identify issues early.
- **Activities:**
    - Continuous monitoring tools like Nagios, Prometheus, and Grafana are used to monitor system performance, application health, and logs.
    - Alerts and notifications are set up to catch errors or issues in real time.
    - Feedback from monitoring informs the next development phase.

8. Operations (Continuous Feedback & Continuous Improvement)

- **Objective:** Continuously improve by learning from the monitoring phase and optimizing the software.
- **Activities:**

- Operations teams analyze performance and user feedback to identify any areas of improvement.
- Issues are addressed promptly, and features are iteratively improved.
- Feedback loops help guide planning, development, and deployment for future releases.

The DevOps life cycle is a continuous and iterative process. It emphasizes collaboration and automation at every phase to deliver high-quality software quickly and efficiently, while adapting to feedback and improvements based on operational data.

Set 3

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

A.

Agile Model:

The agile model is an iterative approach to software development that involves breaking down projects into smaller parts and delivering them in stages. Agile models are based on the idea of continuous improvement and feedback cycles.

Waterfall Model:

The waterfall model is a linear, sequential approach to project management that breaks down development into distinct phases.

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	Testing can only be performed when the complete product is ready.

development of the entire product.	
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 eco system.

A. DevOps Ecosystem: A Detailed Discussion

The **DevOps ecosystem** is a set of practices, tools, and cultural philosophies that combine **software development (Dev)** and **IT operations (Ops)** to enhance collaboration, automation, and integration throughout the software development lifecycle (SDLC). DevOps aims to shorten development cycles, improve software quality, and deliver continuous value to customers.

The DevOps ecosystem involves several key components, tools, practices, and stages that facilitate collaboration between development, operations, and other stakeholders, ensuring a seamless and efficient software delivery process. Here's an in-depth explanation of the DevOps ecosystem:

Key Components of the DevOps Ecosystem

1. **Collaboration and Communication**:
    - One of the core principles of DevOps is the **cultural shift** that focuses on fostering collaboration between the development and operations teams. Traditionally, development teams write code while operations teams handle deployment and maintenance. DevOps breaks down these silos, encouraging shared responsibilities and communication between teams.
    - Tools like **Slack**, **Microsoft Teams**, and **JIRA** are commonly used to enhance communication between developers, testers, and operations teams, allowing for transparent information exchange.
2. **Automation**:
    - Automation is a central concept in DevOps, as it minimizes human intervention and ensures consistency in software deployment, testing, and operations.
    - Key areas where automation is applied in DevOps:
        - **Continuous Integration (CI)**: Developers integrate code into a shared repository multiple times a day.

        - **Continuous Delivery (CD)**: Automatically deploy and release applications to various environments.
        - **Infrastructure as Code (IaC)**: Automates the setup and management of infrastructure using code.
        - **Automated Testing**: Run tests automatically to ensure quality and performance.
    - Tools: **Jenkins**, **GitLab CI/CD**, **Terraform**, **Ansible**, **Chef**, **Puppet**.
3. **Version Control**:
    - A version control system tracks changes in the codebase, allows collaboration among team members, and maintains code history.
    - Git is the most widely used version control system.
    - Tools: **Git**, **GitHub**, **GitLab**, **Bitbucket**.
4. **Continuous Integration (CI)**:
    - CI is the practice of merging all developers' working copies of code into a shared repository frequently (at least once a day).
    - The goal of CI is to detect integration issues as early as possible, reducing the time spent on bug fixing and improving code quality.
    - CI tools automate the process of code integration, build, and initial testing.
    - Tools: **Jenkins**, **CircleCI**, **Travis CI**, **Bamboo**.
5. **Continuous Delivery (CD)**:
    - Continuous Delivery automates the release of software to production with minimal manual intervention. It ensures that the software is always in a deployable state.
    - The primary focus of CD is to deliver changes to users quickly and safely by deploying code changes automatically after they pass through the CI pipeline.
    - Tools: **Jenkins**, **Spinnaker**, **Argo CD**, **Octopus Deploy**.
6. **Infrastructure as Code (IaC)**:
    - IaC refers to the process of managing and provisioning infrastructure through code rather than manual processes. It allows for automation of system configurations, ensuring that the infrastructure is consistent, scalable, and reliable.
    - IaC makes it easier to deploy, scale, and maintain environments in a repeatable and consistent manner.
    - Tools: **Terraform**, **Ansible**, **Chef**, **Puppet**, **CloudFormation**.
7. **Configuration Management**:
    - Configuration management ensures that the system's infrastructure is configured consistently across all environments.
    - It enables automation of the setup, configuration, and maintenance of servers and other infrastructure components.
    - Tools: **Chef**, **Puppet**, **Ansible**, **SaltStack**.
8. **Monitoring and Logging**:
    - Monitoring and logging are critical in DevOps to ensure the health and performance of applications and infrastructure.
    - Monitoring tools track the performance and availability of systems, while logging tools provide detailed logs of system activities, which can be used for debugging and identifying problems.

- Continuous monitoring helps in detecting issues early and taking corrective action before they impact users.
- Tools: **Prometheus**, **Grafana**, **Nagios**, **New Relic**, **Datadog**, **Elasticsearch**, **Kibana**, **Splunk**.

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

A.

To adopt DevOps in IT projects, key steps include: establishing a culture of collaboration between development and operations teams, implementing version control systems like Git, automating build and testing processes, utilizing continuous integration and deployment pipelines, adopting infrastructure as code, containerizing applications, and continuously monitoring performance with feedback loops to identify and address issues rapidly; all while prioritizing a focus on automation and measuring key DevOps metrics to track progress and identify areas for improvement.

Detailed steps:

Cultural Shift:

Break down silos: Foster open communication and collaboration between development and operations teams to break down traditional barriers and encourage shared responsibility.

Embrace a learning culture: Encourage continuous learning and feedback loops to identify and address issues promptly.

Promote shared ownership: Ensure both teams feel accountable for the entire software delivery lifecycle.

Toolchain Selection:

Version control system (VCS): Implement a system like Git to manage code changes effectively, allowing for easy collaboration and rollback options.

Continuous integration (CI) server: Utilize tools like Jenkins, GitLab CI, or CircleCI to automate the build, testing, and integration process on every code change.

Infrastructure as code (IaC): Leverage tools like Terraform or Ansible to define and manage infrastructure through code, enabling consistent deployments across environments.

Containerization platform: Consider Docker to package applications into self-contained units for easier deployment and scaling.

Monitoring tools: Deploy monitoring solutions like Prometheus, Datadog, or New Relic to gather real-time insights into application performance and health.

Implementation of DevOps Practices:

Continuous development (CD): Encourage frequent small code commits and integrate them into the main codebase regularly.

Continuous integration (CI): Automate the process of building, testing, and integrating code changes to identify issues early.

Continuous testing (CT): Implement automated tests at all stages of the development lifecycle to ensure quality throughout.

Continuous delivery (CD): Automate the deployment process to production-like environments, enabling rapid delivery of new features.

Continuous monitoring (CM): Actively monitor applications in production to detect potential issues and quickly respond to incidents.

Automation:

Infrastructure provisioning: Automate the process of setting up and configuring infrastructure using IaC tools.

Deployment process: Automate the deployment of applications to different environments (development, staging, production).

Testing procedures: Automate repetitive testing tasks to improve efficiency.

Feedback Loop:

Metrics collection: Identify key performance indicators (KPIs) like deployment frequency, lead time, and mean time to recover (MTTR) to track progress.

Post-mortem analysis: Regularly review incidents to learn from mistakes and improve processes

Continuous improvement: Use feedback from monitoring and analysis to iterate on DevOps practices and tools

Key Considerations:

Start small: Begin with a pilot project to test and refine DevOps practices before scaling across the organization.

Training and education: Provide training to all team members on DevOps principles and tools.

Leadership support: Secure strong leadership buy-in to drive cultural change and resource allocation for successful DevOps adoption.

Set 4

1.Explain the values and principles of Agile model.

A. 4 Values of the Agile Model:

1. **Individuals and Interactions over Processes and Tools: Focuses on the importance of effective communication and collaboration among team members.**
2. **Working Software over Comprehensive Documentation: Prioritizes the delivery of functional software as the primary measure of progress.**
3. **Customer Collaboration over Contract Negotiation: Encourages customers and stakeholders to have active involvement throughout the development process.**
4. **Responding to Change over Following a Plan: On changing requirements, embracing flexibility and ability to adapt even late in the development process.**

12 Principles of Agile Model:

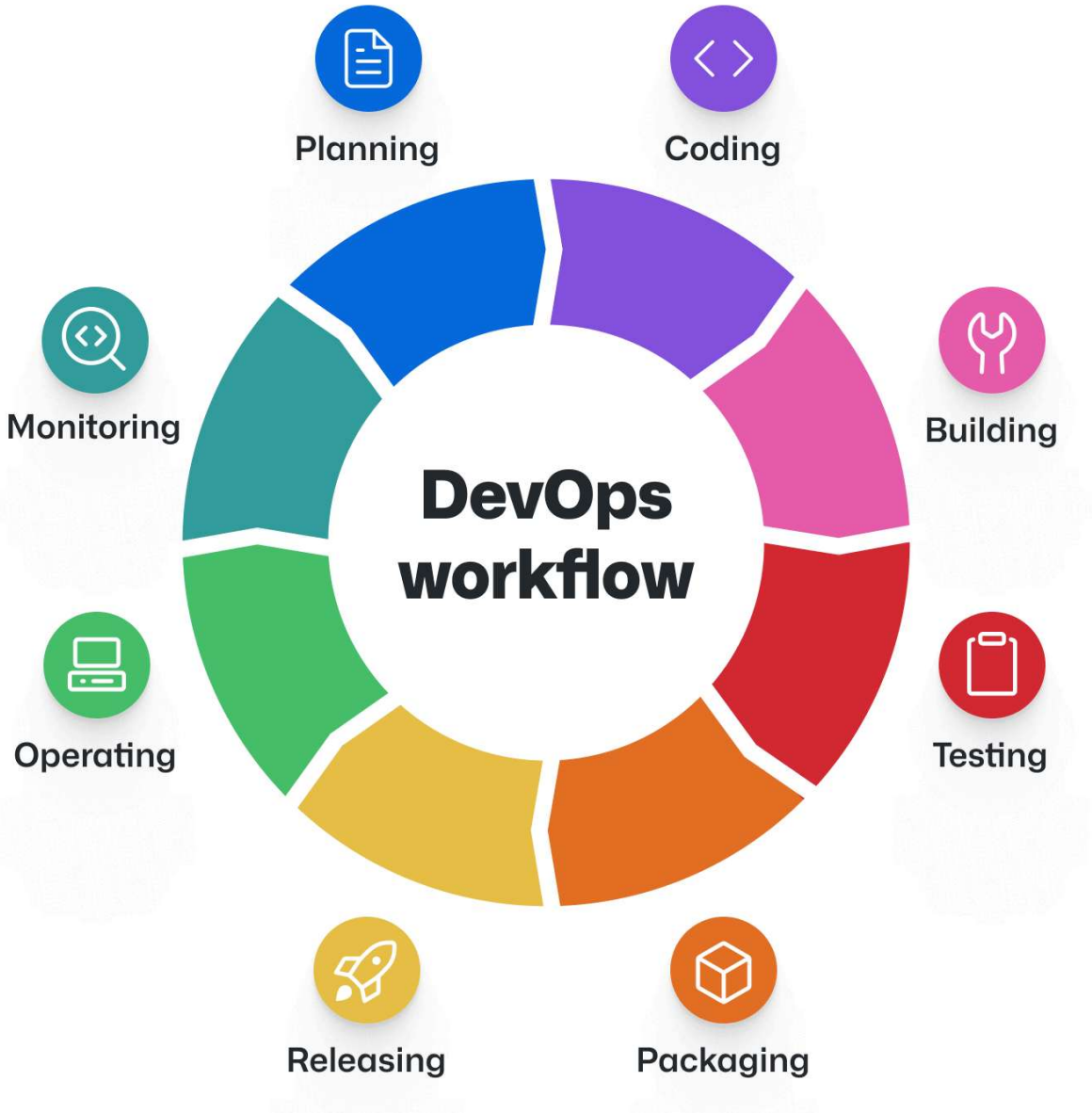
1. **Customer Satisfaction through Early and Continuous Delivery:** This principle concentrates on the importance of customer satisfaction by providing information to customers early on time and also with consistency throughout the development process.
2. **Welcome Changing Requirements, Even Late in Development:** Agile processes tackle change for the customer's competitive advantage. Even late in development, changes in requirements are welcomed to ensure the delivered software meets the evolving requirements of the customer.
3. **Deliver Working Software Frequently:** This principle encourages the regular release of functional software increments in short iterations. This enables faster feedback and adaptation to changing requirements.
4. **Collaboration between Business Stakeholders and Developers:** This says the businesspeople and developers must work together daily throughout the project. There should be communication and collaboration between stakeholders and the development team regularly. This is crucial for understanding and prioritizing requirements effectively.

5. **Build Projects around Motivated Individuals**: This promotes in giving developers the environment and support they need and trusts them to complete the job successfully. Motivated and empowered individuals are more likely to produce work with quality and make valuable contributions to the project.
6. **Face-to-face communication is the Most Effective:** Face-to-face communication is the most effective method of discussion and conveying information. This principle depicts the importance of direct interaction which helps minimize misunderstandings, and hence effective communication is achieved.
7. **Working Software is the Primary Measure of Progress:** This principle emphasizes delivering functional and working software as the primary metric for project advancement. It encourages teams to prioritize the continuous delivery of valuable features, so it ensures that good progress is consistently achieved throughout the process. The primary goal is to provide customers with incremental value and also gather feedback early in the project life cycle.
8. **Maintain a Sustainable Pace of Work:** Agile promotes sustainable development. All people involved: The sponsors, developers, and users should be able to maintain a constant pace indefinitely. This principle depicts the need for a sustainable and consistent development pace. This helps in avoiding burnout and ensures long-term project success.

9. **Continuous Attention to Technical Excellence and Good design:** This principle is on the importance of maintaining high standards of technical craft and design, so it ensures the long-term ability in maintenance and adaptability of the software.
10. **Simplicity—the Art of Maximizing the Amount of Work Not Done:** Simplicity is essential. The objective here is to concentrate on the most valuable features and tasks and avoid unnecessary complexity as the art of maximizing the amount of work not done is crucial.
11. **Self-Organizing Teams:** Self-organizing teams provide the best architectures, requirements, and designs. These help in empowering teams to make decisions and organize to optimize efficiency and creativity.
12. **Regular Reflection on Team Effectiveness:** This makes the team reflect on how to become more effective at regular intervals and then adjust accordingly. Continuous improvement is very crucial for adapting to changing circumstances and optimizing the team's performance over time.

2. Write a short notes on the DevOps Orchestration.

A.DevOps orchestration tames the complexity of DevOps toolchains by automatically managing workflows and dependencies in DevOps workflows. DevOps is a widely practiced set of procedures and tools for streamlining the development, release, and updating of software.



DevOps orchestration essentially acts as a conductor, coordinating and automating multiple tasks across different DevOps tools within a pipeline, effectively managing the complexities of a DevOps toolchain by ensuring smooth workflow and dependency management throughout the software development lifecycle, from coding to deployment.

Key points about DevOps orchestration:

Beyond automation:
While automation focuses on individual tasks, orchestration takes it further by coordinating and sequencing these tasks to achieve a larger goal within the DevOps pipeline.

Workflow management:
It orchestrates the flow of activities across different tools, managing dependencies between them to ensure smooth execution of the entire process.

Streamlining complexity:
By automating and coordinating complex DevOps workflows, orchestration helps teams manage large, multifaceted projects with greater efficiency.

Benefits:

- Faster delivery of software updates
- Reduced manual intervention and potential errors
- Improved collaboration between development and operations teams
- Enhanced visibility into the entire DevOps pipeline.

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

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

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

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

	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.	partitioned among the improvement and operation groups.
9.	Spry accepts “smaller and concise”. Littler the group superior it would be to convey with fewer complexities.	DevOps, on the other hand, accepts that “bigger is better”.
10.	Since Agile groups are brief, a foreordained sum of time is there which are sprints. Tough, it happens that a sprint has endured longer than a month but regularly a week long.	DevOps, on the other hand, prioritizes reliabilities. It is since of this behavior that they can center on a long-term plan that minimizes commerce’s unsettling influences.

11.	A big team for your project is not required.	It demands collaboration among different teams for the completion of work.
12.	**Some of the Tools-** • Bugzilla • JIRA • Kanboard and more.	**Some of the Tools-** • Puppet • Ansible • AWS • Chef • team City OpenStack and more.