DevOps Mid-1 Set-1

1)explain importance of agile software development.[10M]

Agile software development is a methodology that focuses on iterative development,collaboration, and customer satisfaction. It emphasizes flexibility, adaptability, and continuous improvement in the software development lifecycle. **[2M]**

Importance of Agile Software Development in DevOps

Agile and DevOps complement each other, creating a fast, efficient, and adaptive software development lifecycle. Here's why Agile is crucial in a DevOps environment:[1M]

1. Faster Development and Deployment

- Agile promotes iterative development with short sprints, which aligns perfectly with DevOps' goal of continuous integration and continuous delivery (CI/CD).
- Frequent releases ensure faster feedback, reducing the time-to-market for new features. **:[1M]**

2. Collaboration Between Teams

- Agile fosters close collaboration between development, operations, and QA teams, which is essential for DevOps.
- Cross-functional teams improve communication, reducing silos and bottlenecks**. [1M]**

3. Improved Quality and Reliability

- Agile's emphasis on testing (TDD, BDD) and continuous feedback ensures high-quality code.
- DevOps automates testing and deployment, minimizing manual errors and increasing reliability. **[1M]**

4. Continuous Improvement

- Agile encourages adaptability through regular retrospectives and feedback loops.
- DevOps enables real-time monitoring and rapid responses to issues, enhancing system stability. **[1M]**

5. Customer-Centric Approach

- Agile prioritizes user feedback, allowing teams to pivot quickly based on customer needs.
- DevOps ensures that updates reach users faster with minimal downtime**. [1M]**

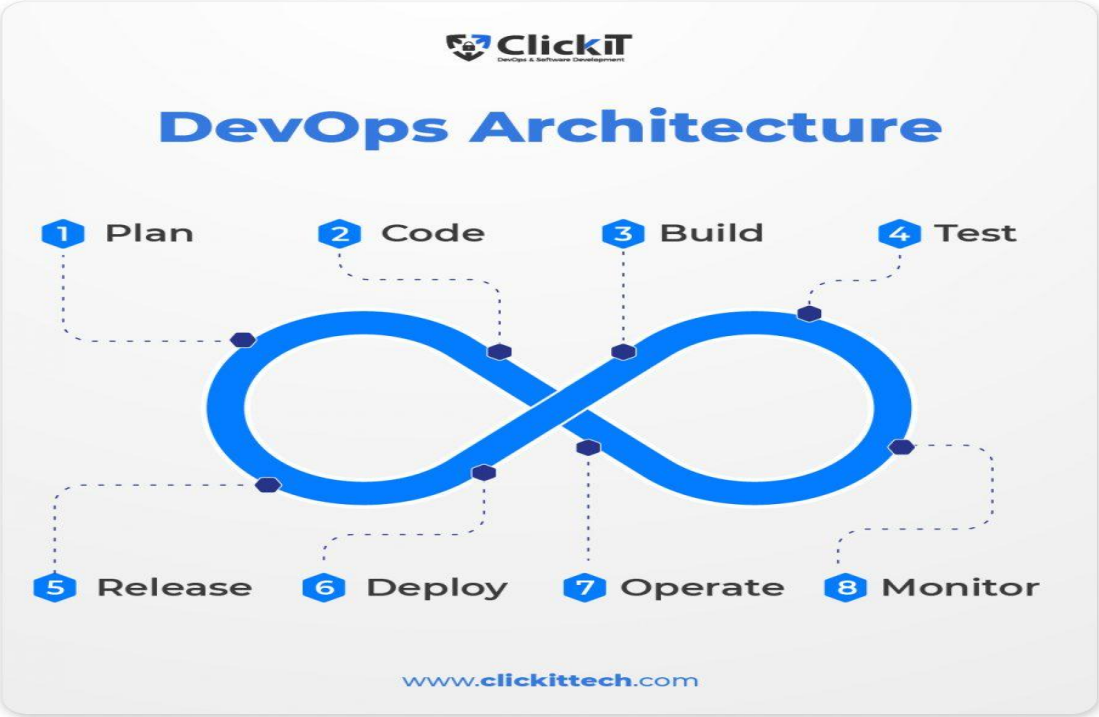
6. Scalability and Flexibility

- Agile allows teams to scale projects efficiently by breaking them into manageable iterations.

- DevOps provides the infrastructure and automation needed to scale deployments seamlessly.[1M]

By integrating Agile methodologies with DevOps practices, organizations can achieve faster delivery, higher-quality software, and better collaboration across teams. **[1M]**

2)explainDevOps architecture and its features with neat sketch.[10M]



DevOps is an approach that combines **development (Dev)** and **operations (Ops)** to streamline software delivery, enhance collaboration, and improve automation. Below is a breakdown of the **8 key phases of DevOps architecture**, explaining their roles, tools, and best practices**.** **[2M]**

1.Plan (Requirement Analysis & Roadmap)

Purpose:

- Define project scope, objectives, and deliverables.
- Prioritize features using Agile methodologies.
- Set milestones and sprint goals.

Key Activities:

 Gather requirements from stakeholders.
 Create user stories and tasks.
 Plan sprints and backlog using Agile methods.

Tools:

 Jira, Trello, Azure DevOps, Confluence **[1M]**

2.Code (Development & Version Control)

Purpose:

- Developers write, review, and commit code.
- Version control tracks changes and enables collaboration.

Key Activities:

 Code writing and peer review.
 Use Git branching strategies (e.g., GitFlow).
 Maintain secure coding practices.

Tools:

 Git, GitHub, GitLab, Bitbucket **[1M]**

3.Build (Continuous Integration & Compilation)

Purpose:

- Convert source code into executable software.
- Resolve dependencies and create build artifacts.

Key Activities:

 Automate builds using CI/CD pipelines.
 Manage dependencies efficiently.
 Detect early build failures.

Tools:

 Jenkins, Maven, Gradle, GitHub Actions **[1M]**

4.Test (Automated Testing & Quality Assurance)

Purpose:

- Ensure the application is bug-free before release.
- Automate various testing levels (unit, integration, security, performance).

Key Activities:

 Run unit tests, integration tests, and regression tests.
 Automate test scripts using frameworks.
 Identify vulnerabilities and security flaws.

Tools:

 Selenium, JUnit, TestNG, Postman, SonarQube **[1M]**

5.Release (Versioning & Deployment Readiness)

Purpose:

- Prepare software for deployment to staging or production.
- Ensure version control, change approvals, and rollback strategies.

Key Activities:

 Define versioning strategies.
 Automate approval workflows.
 Store release artifacts in repositories.

Tools:

 Jenkins, Spinnaker, AWS CodeDeploy, Nexus, Artifactory **[1M]**

6.Deploy (Continuous Deployment & Delivery)

Purpose:

- Deploy applications in production with minimal downtime.
- Ensure rollback mechanisms for failures.

Key Activities:

 Automate deployment pipelines.
 Use blue-green or canary deployments.
 Manage infrastructure as code (IaC).

Tools:

 Docker, Kubernetes, Ansible, Terraform **[1M]**

7.Operate (Infrastructure & Performance Management)

Purpose:

- Ensure system uptime, scalability, and infrastructure management.
- Automate configuration and orchestration.

Key Activities:

 Monitor server health and app performance.
 Scale infrastructure dynamically.
 Apply security patches and updates.

Tools:

 Prometheus, Datadog, AWS CloudWatch **[1M]**

8.Monitor (Logging & Continuous Feedback)

Purpose:

- Monitor system health, application performance, and security threats.
- Provide insights for continuous improvement.

Key Activities:

 Log analysis and root cause detection.
 Automated alerts for anomalies.
 Collect user feedback for optimization.

Tools:

 Splunk, ELK Stack, New Relic, Nagios **[1M]**

3. Describe various features and capabilities in Agile.[10M]

Agile capabilities play a vital role in enabling **DevOps workflows** by fostering **collaboration, flexibility, and continuous improvement**. Below is a breakdown of Agile capabilities under key areas:

1.AgileMindset

Agile is more than just a methodology—it's a **mindset** that emphasizes **adaptability, collaboration, and customer-centric development**.

Key Capabilities:

 Embracing Change: Teams quickly adapt to new requirements.
 Collaboration & Transparency: Open communication between Dev, Ops, and QA.
 Customer Focus: Continuous feedback ensures better product alignment.
 Fail Fast, Learn Fast: Encourages experimentation and innovation.

DevOps Connection:

- The Agile mindset**eliminates silos** by integrating development and operations teams.
- Encourages a **culture of shared responsibility** for faster software delivery**.** **[2M]**

2.Agile Methodologies

Agile methodologies provide structured frameworks for **incremental, iterative software development** in DevOps.

Key Capabilities:

 Scrum: Organizes work into sprints with backlog prioritization.
 Kanban: Focuses on continuous delivery with visualized workflows.
 Lean: Eliminates waste and optimizes processes.
 Extreme Programming (XP): Encourages frequent releases and test-driven development.

DevOps Connection:

- **Scrum &Kanban** help teams manage DevOps pipelines efficiently.
- **Lean principles** align with DevOps automation to reduce delays and waste.

 Tools: Jira, Trello, Azure Boards **[2M]**

3.Agile Practices

Agile practices **enhance collaboration, automation, and efficiency** in DevOps.

Key Capabilities:

 User Stories & Backlog Grooming: Prioritizes customer needs.
 Test-Driven Development (TDD): Automates testing in CI/CD pipelines.
 Pair Programming & Code Reviews: Improves code quality.
 Retrospectives: Drives continuous improvement.

DevOps Connection:

- TDD and automated testing enable **seamless CI/CD workflows**.
- Retrospectives ensure **DevOps processes evolve continuously**.

 Tools: GitHub, GitLab, Jenkins, Selenium **[2M]**

4.Agile Tools

Agile tools facilitate **collaboration, automation, and project management** in DevOps.

Key Capabilities:

 Version Control Tools: Manage source code and track changes.
 CI/CD Tools: Automate builds, testing, and deployments.
 Monitoring & Feedback Tools: Ensure system reliability and performance.
 Collaboration Tools: Enhance team communication and transparency.

DevOps Connection:

- Agile tools **integrate with DevOps pipelines** to streamline development and operations.
- Automation in Agile tools enhances **continuous delivery**.

 Examples:

- **Collaboration:**Jira, Confluence, Slack
- **CI/CD:** Jenkins, GitHub Actions, GitLab CI/CD
- **Monitoring:** Prometheus, Splunk, ELK Stack **[2M]**

5.Continuous Delivery & Continuous Improvement

Continuous Delivery (CD) ensures **frequent, reliable releases**, while Continuous Improvement (CI) drives **process optimization**.

Key Capabilities:

 CI/CD Pipelines: Automate code integration, testing, and deployment.
 Infrastructure as Code (IaC): Manages infrastructure efficiently.
 Automated Monitoring & Feedback: Detects and resolves issues in real-time.
 Retrospectives & Iteration Planning: Helps teams refine workflows.

DevOps Connection:

- Continuous Delivery **accelerates DevOps release cycles**.
- Continuous Improvement ensures **long-term efficiency and innovation**.

 Tools: Kubernetes, Terraform, ArgoCD, New Relic **[2M]**

Set-2

1)What is SDLC?Explain various phases involved in SDLC.[10M]

Software Development Life Cycle (SDLC) Overview

The **Software Development Life Cycle (SDLC)** is a structured process used by development teams to **design, develop, test, and deploy software** efficiently. It ensures **high-quality software** while meeting customer requirements, staying within budget, and reducing risks. **[2M]**

SDLC Phases with Explanation

1.Planning

 Purpose: Define project scope, objectives, and requirements.
 Key Activities:
 Requirement gathering from stakeholders.
 Feasibility analysis (technical, financial, operational).
 Risk assessment and mitigation strategies.
 Defining timelines, costs, and resource allocation.

 Tools: Jira, Trello, Microsoft Project

 Output: Software Requirement Specification (SRS), project roadmap**.** **[1M]**

2.Coding (Development)

 Purpose: Convert design into actual software by writing code.
 Key Activities:
 Developers write, review, and commit code to repositories.
 Follow coding standards and best practices.
 Implement version control for tracking changes.
 Perform unit testing to catch early defects.

 Tools: Git, GitHub, GitLab, Bitbucket, Visual Studio Code

 Output: Source code, executable files. **. [1M]**

3.Building

 Purpose: Convert source code into deployable software.
 Key Activities:
 Compile and package code into executables.
 Resolve dependencies and generate build artifacts.
 Automate builds using CI/CD pipelines.

 Tools: Jenkins, Maven, Gradle, GitHub Actions

 Output: Built application ready for testing. **. [1M]**

4.Testing

 Purpose: Identify and fix defects before production release.
 Key Activities:
 Execute unit, integration, functional, and performance tests.
 Automate testing using CI/CD pipelines.
 Security testing to prevent vulnerabilities.
 User acceptance testing (UAT) for validation.

 Tools: Selenium, JUnit, TestNG, Postman, SonarQube

 Output: Test reports, bug tracking documentation. **. [1M]**

5.Release

 Purpose: Ensure a smooth transition of software to deployment.
 Key Activities:
 Prepare deployment artifacts and versioning.
 Conduct final security and compliance checks.
 Approval process before production rollout.

 Tools: Nexus, Artifactory, AWSCodeDeploy

 Output: Stable, versioned release package. **. [1M]**

6.Deployment

 Purpose: Deploy software to staging or production environments.
 Key Activities:
 Deploy software using automated pipelines.
 Use blue-green, rolling, or canary deployment strategies.
 Ensure zero-downtime releases and rollback plans.

 Tools: Docker, Kubernetes, Ansible, Terraform

 Output: Live software in production. **. [1M]**

7.Operate

 Purpose: Ensure application runs smoothly in production.
 Key Activities:
 Manage infrastructure and system configurations.
 Scale applications based on demand.
 Apply patches and updates as needed.

 Tools: AWS, Azure, Terraform, Prometheus

 Output: Optimized and scalable software environment. **. [1M]**

8.Monitoring

 Purpose: Continuously track application performance and issues.
 Key Activities:
 Monitor logs, user activity, and system health.
 Detect and resolve errors in real-time.
 Collect customer feedback for improvements.

 Tools: ELK Stack, Splunk, New Relic, Datadog

 Output: Performance reports, issue tracking. **. [1M]**

2)Explain briefly about various stages involved in the DevOps pipeline.[10M]

A DevOps pipeline is an automated workflow that integrates development and operations,ensuring continuous software delivery. It includes multiple stages that streamline codedevelopment, testing, and deployment. . **[1M]**

Stages in DevOps Pipeline

1. Plan

Requirements are gathered, and tasks are planned.

Tools: Jira, Confluence, Trello. . **[1M]**

2. Develop

Developers write and commit code.

Version control is used for collaboration.

Tools: Git, GitHub, GitLab, Bitbucket. . **[1M]**

3. Build

Source code is compiled into executable files.

Automated build tools ensure consistency.

Tools: Maven, Gradle. . **[1M]**

4. Test

Automated testing is performed to check software functionality.

Includes unit testing, integration testing, and security testing.

Tools: Selenium, JUnit, TestNG. . **[1M]**

5. Release

The software is packaged and prepared for deployment.

Tools: Docker, Kubernetes. . **[1M]**

6. Deploy

Code is deployed to production or staging environments.

Continuous Deployment (CD) ensures fast releases.

Tools: Jenkins, Ansible, AWS CodeDeploy. . **[1M]**

7. Operate

Monitoring and logging ensure system stability.

Tools: Prometheus, ELK Stack. . **[1M]**

8. Monitor

Performance tracking and error detection.

Provides real-time feedback for future improvements.

Tools: Nagios, Splunk, Datadog. . **[1M]**

Conclusion

The DevOps pipeline automates the software development lifecycle, improving efficiency,reducing errors, and ensuring rapid software delivery. . **[1M]**

3. Describe the phases in the DevOps life cycle. [10M]

Phases in DevOpsLifeCycle:

1. Plan

Requirements are gathered, and tasks are planned.

Tools: Jira, Confluence, Trello. **[1M]**

2. Develop

Developers write and commit code.

Version control is used for collaboration.

Tools: Git, GitHub, GitLab, Bitbucket. **[1M]**

3. Build

Source code is compiled into executable files.

Automated build tools ensure consistency.

Tools: Maven, Gradle. **[1M]**

4. Test

Automated testing is performed to check software functionality.

Includes unit testing, integration testing, and security testing.

Tools: Selenium, JUnit, TestNG. **[1M]**

5. Release

The software is packaged and prepared for deployment.

Tools: Docker, Kubernetes. **[1M]**

6. Deploy

Code is deployed to production or staging environments.

Continuous Deployment (CD) ensures fast releases.

Tools: Jenkins, Ansible, AWS CodeDeploy. **[1M]**

7. Operate

Monitoring and logging ensure system stability.

Tools: Prometheus, ELK Stack. **[1M]**

8. Monitor

Performance tracking and error detection.

Provides real-time feedback for future improvements.

Tools: Nagios, Splunk, Datadog. **[1M]**

Conclusion

The DevOps pipeline automates the software development lifecycle, improving efficiency,reducing errors, and ensuring rapid software delivery. **[2M]**

Set-3

1.Write the difference between Waterfall and Agile models?[10M]

Agile Model	Waterfall Model
Client input is required throughout the	Client input is required only after

Agile Model	Waterfall Model
product development.	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.
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.

[10M]

2. Discuss in detail about DevOps ecosystem?[10M]

The DevOps ecosystem refers to the collection of tools, practices, and cultural philosophies that together enable organizations to implement DevOps principles and practices. DevOps seeks to improve collaboration between development (Dev) and operations (Ops) teams to deliver high-quality software faster and more reliably. The ecosystem encompasses a wide range of tools and technologies that span the entire software development lifecycle (SDLC), from planning and coding to deployment and monitoring**.** **[1M]**

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. **[1M]**

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. **[1M]**

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. **[1M]**

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. **[1M]**

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. **[1M]**

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. **[1M]**

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. **[1M]**

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. **[1M]**

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. **[1M]**

Benefits of DevOps Ecosystem:

 Improves software delivery speed.

 Enhances collaboration between teams.

 Reduces deployment failures and downtime.

 Ensures continuous monitoring and feedback. **[1M]**

3. List and explain the steps followed for adopting DevOps in IT projects.[10M]

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. **[2M]**

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. **[2M]**

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. **[2M]**

Continuous monitoring: Implement real-time monitoring tools to track application performance, identify potential issues, and receive alerts proactively. **[2M]**

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. **[2M]**

SET-4

1. Explain the values and principles of Agile model.[10M]

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. **[1M]**

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. **[1M]**

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. **[1M]**

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. **[1M]**

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**.[1/2 mark]**

2. Welcome Changing Requirements: Embrace changes in requirements, even late in development, as they can lead to better outcomes for the customer. **[1/2mark]**

3. Deliver Working Software Frequently: Aim for shorter timescales between releases, delivering working software every few weeks or months. **.[1/2mark]**

4. Business People and Developers Must Work Together Daily: Foster close collaboration between business stakeholders and developers throughout the project. **.[1/2mark]**

5. Build Projects Around Motivated Individuals: Provide a supportive environment for motivated individuals, trusting them to get the job done. **.[1/2mark]**

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. **.[1/2mark]**

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. **.[1/2 mark]**

8. Sustainable Development: Promote a constant pace of work that can be maintained indefinitely without burnout or stress. **.[1/2 mark]**

9. Technical Excellence and Good Design Enhance Agility: Encourage technical excellence and good design practices to improve agility and adaptability in development efforts. **.[1/2mark]**

10. Simplicity is Essential: Maximize the amount of work not done by focusing on simplicity; this helps streamline processes and reduce complexity. **.[1/2 mark]**

11. Self-Organizing Teams Produce the Best Results: Allow teams to self-organize, fostering creativity and innovation while encouraging ownership of their work. **.[1/2 mark]**

12. Regularly Reflect on How to Become More Effective: Conduct regular retrospectives to reflect on processes and identify opportunities for improvement continuously. **.[1/2mark]**

2. Write a short notes on the DevOps Orchestration.[10M]

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**.** **[1M]**

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**.[2M]**

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. **.[2M]**

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. **.[2M]**

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. **.[2M]**

Benefits of DevOps Orchestration

 Reduced Time to Market: By streamlining processes and automating workflows, organizations can accelerate their software delivery timelines, allowing them to respond quickly to market demands.

 Improved Efficiency: Orchestration minimizes manual intervention in repetitive tasks, reducing human error and freeing up teams to focus on more strategic initiatives.

 Enhanced Collaboration: By integrating various tools and processes, orchestration fosters better communication among development, operations, and other stakeholders.

 Increased ROI: Effective orchestration maximizes the return on investment in automation tools by ensuring they are utilized efficiently across the organization. **[1M]**

3.What is the difference between Agile and DevOps models?[10M]

S. No.	Agile	DevOps
1.	It started in the year 2001.	It started in the year 2007.

S. No.	Agile	DevOps
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 can loan help instead of holding up for the group leads or any pro impedances.	DevOps features a diverse approach and is very viable, most of the time it takes after “Divide and Conquer”. Work partitioned among the improvement and operation groups.
9.	Spry accepts “smaller and concise”. Littler the group superior it would be to convey with fewer complexities.	DevOps, on the other hand, accepts that “bigger is better”.
10.	Since Agile groups are brief, a foreordained sum of time is there which are sprints. Tough,	DevOps, on the other hand, prioritizes reliabilities. It is since

S. No.	Agile	DevOps
	it happens that a sprint has endured longer than a month but regularly a week long.	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.

[10M]