Name:- Kunal Bhat
Roll No:- CE023
Identity :- 22CEUOG155

1]What is DevOps?

DevOps is a philosophy and set of practices that aims to bridge the gap between development (Dev) and operations (Ops) teams in software development. Traditionally, these teams have operated in silos, leading to inefficiencies, delays, and communication breakdowns. DevOps seeks to break down these barriers by fostering collaboration and automation throughout the entire software development lifecycle (SDLC).Lifecycle of DevOps

2]Why DevOps is Used ?

- Collaboration: DevOps emphasises breaking down silos and fostering a collaborative culture between Dev and Ops teams. This allows for better communication, shared goals, and a more holistic understanding of the software development process.

- Automation: A core principle of DevOps is automating repetitive tasks throughout the SDLC. This includes tasks like building, testing, deploying,and monitoring applications. Automation reduces manual errors,improves efficiency, and frees up developers and operations personnel to focus on higher-value activities.

- Continuous Integration (CI): CI is a practice where developers frequently merge their code changes into a central repository, triggering automated builds and tests. This allows for early detection and correction of bugs,preventing them from accumulating and causing bigger issues later.

- Continuous Delivery (CD): CD extends CI by automating the deployment process. Once code passes the CI pipeline, it can be automatically deployed to production

environments. This enables faster and more frequent deployments, allowing developers to deliver features and fixes to users quicker

- Monitoring and Feedback: Monitoring applications in production is crucial for identifying and resolving issues proactively. DevOps promotes using monitoring tools to gather feedback on application performance and health.

3]DevOps v/s Agile

DevOps

DevOps purpose is to manage end to end engineering processes.It focuses on constant testing and delivery.DevOps focuses on operational and business readiness. Feedback comes from the internal team.

Agile

The agile purpose is to manage complex projects.It focuses on constant testing and delivery. It focuses on constant changes.

Agile focuses on functional and non-functional readiness. In Agile, feedback is coming from the customer.

4] Describe the lifecycle of DevOps.

The DevOps lifecycle typically includes the following stages:

- Plan: Define objectives, requirements, and resources for software development.
- Develop: Create and code software features.
- Build: Compile and integrate code into executable artifacts.
- Test: Verify software functionality and performance through automated testing.
- Deploy: Release software to production environments.
- Operate: Monitor and manage deployed applications, addressing issues as they arise.
- Monitor: Collect and analyze feedback and performance metrics to continuously improve the software.

5]Which tools can be used to apply DevOps?

- Version Control Systems (VCS): Git
- Continuous Integration (CI): Jenkins, Travis CI
- Configuration Management: Ansible, Puppet, Chef
- Containerization and Orchestration: Docker, Kubernetes
- Continuous Deployment and Delivery (CD): Spinnaker, GitLab CI/CD
- Monitoring and Logging: Prometheus, ELK Stack
- Infrastructure as Code (IaC): Terraform, AWS CloudFormation ● Collaboration and Communication: Slack, Microsoft Teams

6]Mention the advantages and disadvantages of DevOps.

Benefits of DevOps

- Faster deployments: By automating tasks and streamlining workflows, DevOps enables faster and more frequent deployments, accelerating software delivery.

- Improved quality: Early and frequent testing with CI/CD leads to higher quality software by identifying and fixing bugs earlier in the development cycle.

- Reduced costs: Automating tasks and minimising manual errors reduces operational costs associated with software development. • Increased innovation: By freeing up developers from repetitive tasks, DevOps allows them to focus on innovation and development of new features.

- Enhanced collaboration: DevOps fosters a culture of collaboration between Dev and Ops teams, leading to better communication and problem-solving.

Drawbacks of DevOps

- Increased complexity: While DevOps aims to simplify workflows, it can initially introduce complexity due to the adoption of new tools, processes, and potentially unfamiliar technologies. This learning curve can be steep and require investment in training and support.

- Focus on speed over quality: Emphasis on faster deployments might lead to prioritising speed over thorough testing and quality assurance. This can introduce potential risks of deploying buggy or unstable software to production environments.

- Vendor lock-in: Reliance on specific DevOps tools and platforms can lead to vendor lock-in, making it difficult and costly to switch to alternative solutions in the future.

7]What is Jenkins ?

- Open source-solution comprising an automation server to enable continuous integration and continuous delivery (CI/CD).

- Automating various stages of software development such as build, test and deployment

8]Why Jenkins is used?

- Many developer working on same repository. Merging of their commit in code repository causes issues in integration.It results in delay in testing and in release of software.
- Developers had to wait till software code was built and tested to check for Errors

9]What are the applications of Jenkins?

a]Easy Installation

- Platform-agnostic:- Jenkins is written in Java, making it independent of the underlying operating system. You can install it on Windows, macOS,Linux, and other operating systems that support Java.

- Self-contained:- Jenkins is a single package containing all necessary components, making installation straightforward.

b]Easy Configuration

- Web interface:- Jenkins provides a user-friendly web interface for configuration and management. This interface allows users to manage jobs, configure plugins, and monitor builds without requiring extensive command-line expertise.

- On-the-fly error checks:- The web interface offers real-time feedback during configuration, highlighting potential errors and providing guidance for correction.

c]Plugins

- Extensive library:- Jenkins provides a vast library of over 1,700 plugins,each offering specific functionalities.

- Customisation:- Plugins enable users to customise Jenkins to fit their specific needs. They can add functionalities not available in the core installation, extending its capabilities and adapting it to diverse development workflows.

d]Extensible

- Scripting support:- Jenkins allows customisation through scripting languages like Groovy and Python. This empowers users to tailor Jenkins behaviour to their specific requirements by writing custom scripts and extending existing functionalities.

- API integration:- Jenkins offers an extensive API that allows integration with external tools and services.

e]Distributed Builds

- Master-slave architecture:- Jenkins employs a master-slave architecture for distributed builds. The master node manages the build process, while slave nodes execute build tasks. This allows for scaling the build infrastructure and running builds in parallel across multiple machines, significantly reducing overall build times for complex projects.

- Dynamic slave allocation:- Jenkins can dynamically allocate slave nodes based on available resources and workload, optimising resource utilisation and ensuring efficient build execution.

10]How to install and use Jenkins for different purposes?

Installation:

- Prerequisites: Install Java Development Kit (JDK).
- Download Jenkins: Get the latest stable release for your OS from the Jenkins website.
- Install Jenkins: Follow installation instructions for your OS.
- Start Jenkins: Start Jenkins using appropriate commands.
- Access Jenkins: Open a web browser and navigate to http://localhost:8080

Using Jenkins:

- Create Jobs: Define tasks like building, testing, and deploying projects.
- Configure Jobs: Set up source code management, triggers, build steps, and post-build actions.
- Run Jobs: Trigger jobs manually or automatically based on events.
- View Build Results: Monitor progress, view console output, test results, and artifacts.
- Extend with Plugins: Install additional plugins to extend functionality.
- Integrate with Other Tools: Integrate Jenkins with version control, issue trackers, and deployment servers.