100 Terms & Services

which every DevOps Engineer should be aware of before working on real time projects

1. Continuous Integration (CI): Automates the process of integrating code changes, ensuring that new code is regularly and automatically tested. Use it to catch integration issues early in the development process.

2. Continuous Deployment (CD): Automates the release of well-tested code changes into production. Employ it to streamline and expedite the delivery of new features and improvements to end-users.

3. Version Control System (VCS): Manages changes to source code over time, enabling collaboration among developers and tracking code history. Essential for team-based development to maintain code integrity.

4. Git: A distributed version control system that allows multiple developers to work on a project concurrently. Widely used for version control, branching, and collaboration in software development.

5. Jenkins: An open-source automation server that facilitates building, testing, and deploying code. Ideal for implementing CI/CD pipelines to automate the software delivery process.

6. Build Automation: Automates the compilation and build processes, ensuring consistency in generating executable code. Use it to save time, reduce errors, and enhance the efficiency of the build phase.

7. Artifact: A deployable unit (e.g., JAR or WAR files) produced during the build process. Helps in packaging and distributing software components for deployment.

8. Maven: A build and project management tool that simplifies the build process, manages dependencies, and creates standardized project structures. Commonly used for Java projects.

9. Gradle: A build automation tool that supports multiple languages and provides flexibility in defining build scripts. Suitable for building projects with diverse language dependencies.

10. Containerization: Involves packaging an application and its dependencies into a container for consistent deployment across different environments. Useful for ensuring consistency and portability.

11. Docker: A platform for containerization, enabling developers to build, ship, and run applications consistently across various environments. Great for streamlining deployment and scaling.

12. Kubernetes: An open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Ideal for managing microservices at scale.

13. Orchestration: Automated coordination and management of multiple components in a system. Employ it to streamline complex workflows, ensuring seamless interaction between services.

14. Microservices: Architectural design approach where a complex application is broken down into smaller, independently deployable services. Suitable for enhancing scalability, flexibility, and maintainability.

15. Infrastructure as Code (IaC): Managing and provisioning infrastructure using code instead of manual processes. Enables automated, repeatable, and consistent infrastructure deployment.

16. Terraform: An IaC tool for provisioning and managing infrastructure across various cloud providers. Use it for declarative configuration and efficient resource provisioning.

17. Ansible: An open-source automation tool for configuration management, application deployment, and task automation. Effective for simplifying repetitive tasks and ensuring consistency.

18. Chef: A configuration management tool that automates the deployment and management of infrastructure. Useful for ensuring the desired state of systems.

19. Puppet: A configuration management tool for automating the provisioning and management of infrastructure. Suitable for maintaining a consistent and scalable IT infrastructure.

20. Configuration Management: Automation of system configuration to maintain consistency and ensure desired states across servers. Essential for managing large-scale and dynamic infrastructures.

21. Monitoring: Observing and collecting data about the performance and health of systems. Critical for identifying issues proactively and ensuring optimal system operation.

22. Alerting: Notifying relevant parties when predefined conditions or thresholds are breached. Essential for quickly responding to incidents and minimizing downtime.

23. Logging: Recording events, errors, and other information generated by applications and systems. Crucial for debugging, troubleshooting, and auditing.

24. ELK Stack (Elasticsearch, Logstash, Kibana): A set of tools for log management and analytics. Useful for collecting, processing, and visualizing log data.

25. Prometheus: An open-source monitoring and alerting toolkit designed for reliability and scalability. Ideal for collecting and querying metrics from diverse systems.

26. Grafana: A platform for monitoring and observability, providing visualization and analytics for various data sources. Effective for creating dashboards to monitor system performance.

27. Application Performance Monitoring (APM): Tools and practices for monitoring and optimizing the performance of applications. Essential for ensuring a positive user experience.

28. Load Balancing: Distributing network traffic across multiple servers to optimize resource utilization and prevent overload. Ensures high availability and improved performance.

29. Reverse Proxy: A server that handles requests on behalf of another server. Enhances security, load balancing, and simplifies the architecture.

30. NGINX: A web server and reverse proxy server that excels at handling high concurrent connections. Suitable for improving web server performance and security.

31. Apache: A widely-used open-source web server. Versatile and suitable for serving dynamic and static content.

32. Serverless Architecture: An approach where developers focus on writing code without managing server infrastructure. Ideal for event-driven and scalable applications.

33. AWS Lambda: A serverless compute service by AWS, executing code in response to events. Suitable for building scalable and cost-effective applications.

34. Azure Functions: Serverless compute service in Microsoft Azure. Enables building event-driven applications with minimal infrastructure management.

35. Google Cloud Functions: Serverless compute service in Google Cloud. Allows executing code in response to events without managing servers.

36. Infrastructure Orchestration: Automates the deployment and configuration of infrastructure components. Ensures efficient and consistent resource provisioning.

37. AWS CloudFormation: AWS service for provisioning and managing AWS infrastructure as code. Ideal for automating resource management in AWS environments.

38. Azure Resource Manager (ARM): Microsoft Azure's IaC service for deploying and managing Azure resources. Simplifies resource lifecycle management.

39. Google Cloud Deployment Manager: Google Cloud's IaC service for managing and deploying resources. Enables repeatable and automated infrastructure deployments.

40. Continuous Testing: An approach that emphasizes ongoing automated testing throughout the software development lifecycle. Ensures early detection of defects and reliable software releases.

41. Unit Testing: Testing individual units or components of code to ensure they function correctly in isolation. Essential for validating the correctness of individual code units.

42. Integration Testing: Testing interactions between different components or systems. Ensures that integrated components work together as intended.

43. System Testing: Testing the entire software system to ensure it meets specified requirements. Essential for validating the overall functionality and performance.

44. Performance Testing: Evaluating the responsiveness, speed, and scalability of a system under different conditions. Identifies performance bottlenecks and potential issues.

45. Security Testing: Assessing a system for vulnerabilities and ensuring it meets security requirements. Essential for identifying and addressing security weaknesses.

46. DevSecOps: Integrating security practices into the DevOps process. Ensures security considerations are part of the entire software development lifecycle.

47. Code Review: The systematic examination of code by developers to identify and fix issues. Enhances code quality and ensures adherence to coding standards.

48. Static Code Analysis: Analyzing source code without executing it to identify potential issues. Helps catch coding errors and security vulnerabilities early in development.

49. Dynamic Code Analysis: Analyzing code during runtime to identify vulnerabilities and assess performance. Provides insights into the behavior of running applications.

50. Dependency Management: Managing external libraries and components that a project relies on. Ensures consistency and security in handling project dependencies.

51. Artifact Repository: A centralized location for storing and managing software artifacts. Facilitates version control and distribution of artifacts.

52. Nexus: A repository manager that simplifies the management of binary components. Ensures efficient artifact storage, retrieval, and versioning.

53. JFrog Artifactory: A universal artifact repository manager supporting various package formats. Enables efficient artifact management and distribution.

54. Continuous Monitoring: Real-time observation of systems and applications for performance and security. Provides immediate insights into system behavior.

55. Incident Response: A planned approach to addressing and managing security incidents. Essential for minimizing the impact of security breaches.

56. Site Reliability Engineering (SRE): A discipline that applies software engineering principles to enhance system reliability and performance.

57. Collaboration Tools: Platforms that facilitate communication and collaboration among team members. Enhance teamwork and knowledge sharing.

58. Slack: A popular team messaging platform that facilitates real-time communication and collaboration among team members.

59. Microsoft Teams: A collaboration platform that integrates chat, video conferencing, and file sharing. Ideal for remote and distributed teams.

60. ChatOps: Integrating chat platforms into development and operations workflows. Enables collaborative decision-making and automation through chat.

61. Versioning: Managing and assigning versions to code, documents, or other artifacts. Enables tracking changes and maintaining a historical record.

62. Semantic Versioning (SemVer): A versioning convention that communicates the nature of changes in software. Enhances compatibility and predictability.

63. Feature Toggles: Flags that enable or disable specific features in a system. Facilitates controlled feature rollouts and experimentation.

64. Blue-Green Deployment: A deployment strategy that reduces downtime by switching between two identical environments. Ensures seamless updates and rollback capabilities.

65. Canary Deployment: A deployment technique that releases new features to a subset of users. Allows testing in a production-like environment before full release.

66. Rolling Deployment: Gradual deployment of changes across servers or instances. Minimizes downtime and ensures a smooth transition.

67. Infrastructure Monitoring: Continuous observation of the health and performance of infrastructure components. Essential for maintaining optimal system operation.

68. Service Level Agreement (SLA): An agreement defining the expected level of service between a provider and a customer. Sets performance expectations and consequences for breaches.

69. Service Level Indicator (SLI): A specific metric or measurement used to track the performance of a service. Essential for quantifying service quality.

70. Service Level Objective (SLO): A target level of performance or reliability for a service. Sets measurable goals for service quality.

71. Chaos Engineering: A practice of intentionally injecting failures into a system to test its resilience and identify weaknesses. Improves system robustness and reliability.

72. GitLab: A web-based Git repository manager that provides CI/CD tools and project management. Offers an integrated platform for the entire DevOps lifecycle.

73. Bitbucket: A Git repository management solution by Atlassian, providing code collaboration and CI/CD capabilities. Ideal for teams using other Atlassian products.

74. Artifact Signing: Verifying the authenticity and integrity of artifacts by attaching cryptographic signatures. Enhances security and ensures trust in software distribution.

75. Secrets Management: Securely storing, managing, and distributing sensitive information, such as API keys and passwords. Essential for protecting confidential data.

76. HashiCorp Vault: A tool for managing secrets and protecting sensitive data. Enables secure storage, access control, and encryption of secrets.

77. Continuous Feedback: The ongoing exchange of information within a team to improve processes and performance. Fosters a culture of continuous improvement.

78. Post-Mortem Analysis: A structured review of incidents to identify causes and prevent recurrence. Enables learning from failures and improving system resilience.

79. Infrastructure Cost Management: Monitoring and controlling costs associated with cloud infrastructure usage. Essential for optimizing expenses and preventing overspending.

80. Cloud Billing: The process of charging users for cloud services based on usage. Informs users about the costs incurred for utilizing cloud resources.

81. Immutable Infrastructure: A deployment model where infrastructure components are never modified after creation. Enhances predictability, reliability, and security.

82. Zero Trust Security Model: A security approach that distrusts both internal and external networks. Requires verification and authorization for every access attempt.

83. Authentication: Verifying the identity of users, systems, or devices attempting to access resources. Ensures secure access to protected assets.

84. Authorization: Granting or denying access permissions to authenticated users or systems. Controls what actions users can perform within a system.

85. Single Sign-On (SSO): A system that allows users to access multiple applications with a single set of credentials. Enhances user experience and security.

86. LDAP: Lightweight Directory Access Protocol, used for accessing and managing directory information. Commonly used for authentication in enterprise environments.

87. OAuth: An authorization framework for granting limited access to a third party without exposing credentials. Enables secure API authorization.

88. RBAC (Role-Based Access Control): Access control based on predefined roles and permissions. Enhances security by ensuring users have appropriate access levels.

89. VPN (Virtual Private Network): A secure network connection that enables users to access resources over the internet securely. Essential for remote and secure communications.

90. Network Security Groups (NSG): Azure's security feature for controlling inbound and outbound traffic to resources. Enhances network security in Azure.

91. Firewall: A security barrier that monitors and controls incoming and outgoing network traffic. Essential for protecting systems from unauthorized access.

92. Server Hardening: The process of securing a server by reducing vulnerabilities and strengthening security configurations. Essential for minimizing the risk of attacks.

93. Distributed Tracing: Tracking and monitoring the flow of requests as they traverse through different services. Essential for identifying performance bottlenecks and latency issues.

94. Jaeger: An open-source, end-to-end distributed tracing system. Enables monitoring and troubleshooting of microservices-based architectures.

95. OpenTelemetry: An observability framework for generating, collecting, and managing telemetry data. Facilitates monitoring and tracing in distributed systems.

96. API Gateway: A server that acts as an entry point for an API, handling tasks such as authentication, authorization, and rate limiting. Streamlines API management and enhances security.

97. CICD Pipeline: A series of automated steps for building, testing, and deploying code changes. Enhances the speed and reliability of the software delivery process.

98. Server Monitoring: Continuous observation of server performance metrics to identify and address issues. Ensures optimal server operation and reliability.

99. Immutable Deployment: Replacing entire instances with new ones instead of modifying existing instances. Enhances consistency and reliability in deployment.

100. Server Provisioning: The process of setting up and configuring servers for use. Essential for ensuring that servers are ready to handle specific workloads.

Hope you find this document helpful for your Azure Learning.

For more such content you can check : https://techyoutube.com/

Now, to Support, just follow me on below socials:

Telegram: https://t.me/LearnDevOpsForFree

Twitter: https://twitter.com/techyoutbe

Youtube: https://www.youtube.com/@T3Ptech