1. **Continuous Integration (CI):**
    - **Meaning:** The practice of automatically integrating code changes from multiple contributors into a shared repository multiple times a day.
2. **Continuous Deployment (CD):**
    - **Meaning:** The automated process of deploying code changes to production environments after passing automated tests in a CI/CD pipeline.
3. **Infrastructure as Code (IaC):**
    - **Meaning:** Managing and provisioning infrastructure through machine-readable script files, enabling automation and consistency.
4. **Configuration Management:**
    - **Meaning:** The process of managing and maintaining the state of infrastructure components through code or automation tools.
5. **Containerization:**
    - **Meaning:** Packaging an application and its dependencies into a container to ensure consistency across different environments.
6. **Microservices:**
    - **Meaning:** Architectural style where an application is composed of small, independent, and loosely coupled services.
7. **Orchestration:**
    - **Meaning:** Coordinating and managing multiple containers or services to work together seamlessly.
8. **DevSecOps:**
    - **Meaning:** Integrating security practices into the DevOps workflow to ensure security is prioritized throughout the development lifecycle.
9. **Git:**
    - **Meaning:** A distributed version control system widely used for source code management and collaboration.
10. **Jenkins:**
    - **Meaning:** An open-source automation server used for building, testing, and deploying code.
11. **Artifact Repository:**
    - **Meaning:** A centralized location for storing and managing binary artifacts generated during the software development process.
12. **Scalability:**
    - **Meaning:** The ability of a system to handle increased workloads or growing data without compromising performance.
13. **Elasticity:**
    - **Meaning:** The ability of a system to automatically scale resources up or down based on demand.
14. **Blue-Green Deployment:**
    - **Meaning:** A deployment strategy where two identical environments (blue and green) are alternately used for production, enabling seamless updates.

15. **Rollback:**
    - **Meaning:** Reverting a system to a previous state or version in case of issues with a new deployment.
16. **Infrastructure Monitoring:**
    - **Meaning:** Observing and tracking the performance and health of infrastructure components.
17. **Log Aggregation:**
    - **Meaning:** Collecting and consolidating log data from various sources for analysis and troubleshooting.
18. **Kubernetes:**
    - **Meaning:** An open-source container orchestration platform for automating the deployment, scaling, and management of containerized applications.
19. **Immutable Infrastructure:**
    - **Meaning:** Treating infrastructure as unchangeable and recreating it entirely rather than modifying existing components.
20. **Zero Downtime Deployment:**
    - **Meaning:** Deploying updates or changes without causing interruptions or downtime for end-users.
21. **GitLab:**
    - **Meaning:** A web-based Git repository manager that provides CI/CD pipelines and other DevOps features.
22. **Pipeline as Code:**
    - **Meaning:** Defining and managing CI/CD pipelines using code to ensure consistency and version control.
23. **Load Balancing:**
    - **Meaning:** Distributing incoming network traffic across multiple servers to ensure even resource utilization and prevent overload.
24. **Docker:**
    - **Meaning:** A platform for developing, shipping, and running applications in containers.
25. **Chef:**
    - **Meaning:** A configuration management tool for automating the deployment and management of infrastructure.
26. **Puppet:**
    - **Meaning:** An open-source configuration management and automation tool for managing infrastructure as code.
27. **Ansible:**
    - **Meaning:** An open-source automation tool used for configuration management, application deployment, and task automation.
28. **Serverless Computing:**
    - **Meaning:** A cloud computing model where cloud providers manage infrastructure, allowing developers to focus on writing code without managing servers.
29. **Monolithic Architecture:**

- **Meaning:** An architecture where all components of an application are tightly integrated into a single codebase.

30. **Infrastructure Provisioning:**
    - **Meaning:** Automatically creating and configuring infrastructure resources using tools like Terraform or CloudFormation.
31. **Fault Tolerance:**
    - **Meaning:** The ability of a system to continue functioning even in the presence of failures or errors.
32. **Greenfield Project:**
    - **Meaning:** A new project or application built from scratch without any legacy constraints.
33. **Distributed Systems:**
    - **Meaning:** A system that consists of multiple independent components or services working together.
34. **Chaos Engineering:**
    - **Meaning:** Intentionally injecting faults or disruptions into a system to identify weaknesses and improve resilience.
35. **SLA (Service Level Agreement):**
    - **Meaning:** A commitment defining the expected level of service between a service provider and a customer.
36. **Monitoring as Code:**
    - **Meaning:** Defining and managing monitoring configurations using code for consistency and version control.
37. **Trunk-Based Development:**
    - **Meaning:** A development approach where developers work on a single branch (trunk), promoting continuous integration.
38. **Feature Toggle:**
    - **Meaning:** A development technique to enable or disable specific features in an application during runtime.
39. **Dark Launching:**
    - **Meaning:** Gradually rolling out new features or changes to a subset of users before full deployment.
40. **Infrastructure Decomposition:**
    - **Meaning:** Breaking down monolithic infrastructure into smaller, more manageable components.
41. **Collaborative Documentation:**
    - **Meaning:** Creating and maintaining documentation collaboratively within the development team.
42. **Post-Mortem Analysis:**
    - **Meaning:** A thorough review and analysis of an incident or outage to identify root causes and prevent future occurrences.
43. **DevOps Culture:**

- **Meaning:** A cultural shift emphasizing collaboration, communication, and shared responsibility between development and operations teams.

44. **Change Management:**
    - **Meaning:** The process of controlling and managing changes to infrastructure or code to prevent disruptions.
45. **Immutable Deployment:**
    - **Meaning:** Deploying updates by replacing existing instances rather than modifying them.
46. **Secrets Management:**
    - **Meaning:** Securely storing, managing, and distributing sensitive information such as passwords and API keys.
47. **GitFlow:**
    - **Meaning:** A branching model for Git that defines a set of rules for managing branches in a project.
48. **Swagger/OpenAPI:**
    - **Meaning:** A specification for building APIs, allowing both humans and computers to understand the capabilities of a service.
49. **Technical Debt:**
    - **Meaning:** The metaphorical concept of accumulated work that needs to be done later due to shortcuts or quick solutions taken during development.
50. **Observability:**
    - **Meaning:** The ability to understand and measure the internal state of a system by analyzing its outputs.
51. **Docker:**
    - **Meaning:** An open-source platform for containerization that simplifies the deployment and scaling of applications.
52. **Kubernetes:**
    - **Meaning:** An open-source container orchestration platform for automating the deployment, scaling, and management of containerized applications.
53. **Scalability:**
    - **Meaning:** The ability of a system to handle increased loads by adding resources or optimizing performance.
54. **Load Balancing:**
    - **Meaning:** Distributing incoming network traffic across multiple servers to ensure optimal resource utilization and prevent overload.
55. **Version Control:**
    - **Meaning:** Managing changes to source code or other documents, allowing collaboration, tracking modifications, and maintaining a version history.
56. **Git:**
    - **Meaning:** A distributed version control system widely used for source code management.

57. **Jenkins:**
    - **Meaning:** An open-source automation server used for building, testing, and deploying code.
58. **Artifact:**
    - **Meaning:** A deployable unit generated during the build process, such as a JAR or WAR file.
59. **Pipeline:**
    - **Meaning:** A series of automated steps that code changes go through, typically including building, testing, and deployment.
60. **Blue-Green Deployment:**
    - **Meaning:** A deployment strategy where two identical environments, "blue" and "green," are used to minimize downtime during updates.
61. **Immutable Infrastructure:**
    - **Meaning:** Treating infrastructure as unchangeable, with updates achieved by replacing existing instances rather than modifying them.
62. **Orchestration:**
    - **Meaning:** Coordinating and managing multiple automated tasks to achieve a specific outcome, often used in the context of container orchestration.
63. **Monitoring:**
    - **Meaning:** Observing and collecting data about the performance and health of systems, applications, and infrastructure.
64. **Alerting:**
    - **Meaning:** Notifying relevant parties when specific conditions or thresholds are met, helping to address issues promptly.
65. **Incident Response:**
    - **Meaning:** A set of processes and activities to detect, respond to, and recover from incidents or outages.
66. **Post-Mortem:**
    - **Meaning:** A retrospective analysis of an incident, focusing on identifying root causes and preventive measures.
67. **Capacity Planning:**
    - **Meaning:** Forecasting and managing computing resources to meet current and future demands.
68. **HAProxy:**
    - **Meaning:** An open-source load balancer and proxy server that distributes incoming traffic across multiple servers.
69. **Firewall:**
    - **Meaning:** A network security device that monitors and controls incoming and outgoing traffic based on predetermined security rules.
70. **Secrets Management:**
    - **Meaning:** Securely storing, managing, and distributing sensitive information such as API keys and passwords.

71. **Ansible:**
    - **Meaning:** An open-source automation tool used for configuration management, application deployment, and task automation.
72. **Chef:**
    - **Meaning:** A configuration management tool that automates the deployment and management of infrastructure.
73. **Puppet:**
    - **Meaning:** An open-source configuration management tool for automating the provisioning and management of infrastructure.
74. **Terraform:**
    - **Meaning:** An open-source IaC tool used for provisioning and managing infrastructure as code.
75. **Serverless Computing:**
    - **Meaning:** A cloud computing model where cloud providers automatically manage the infrastructure, allowing developers to focus on writing code.
76. **Elasticsearch:**
    - **Meaning:** An open-source search and analytics engine commonly used for logging and data analysis.
77. **Log Aggregation:**
    - **Meaning:** Collecting and centralizing logs from multiple sources for analysis and troubleshooting.
78. **Prometheus:**
    - **Meaning:** An open-source monitoring and alerting toolkit designed for reliability and scalability.
79. **Grafana:**
    - **Meaning:** An open-source analytics and monitoring platform used to visualize and analyze data.
80. **Load Testing:**
    - **Meaning:** Evaluating a system's performance under expected load conditions to identify bottlenecks and ensure scalability.
81. **Failover:**
    - **Meaning:** Automatically redirecting traffic to a standby server or system in case of a failure.
82. **Latency:**
    - **Meaning:** The time delay between the initiation of a request and the receipt of the response.
83. **API Gateway:**
    - **Meaning:** A server that acts as an API front-end, receiving requests, enforcing throttling, and routing them to the appropriate Microservices.
84. **Secret Rotation:**
    - **Meaning:** Periodically updating sensitive information, such as passwords or encryption keys, to enhance security.

85. **Chaos Engineering:**
    - **Meaning:** Testing the resilience of a system by intentionally introducing disruptions to identify weaknesses and improve overall reliability.
86. **Dependency Management:**
    - **Meaning:** Managing and tracking dependencies between different components or libraries in a software project.
87. **Immutable Deployment:**
    - **Meaning:** Deploying applications by replacing the entire infrastructure, ensuring consistency and reliability.
88. **Pod:**
    - **Meaning:** The smallest deployable unit in a containerized environment like Kubernetes, containing one or more containers.
89. **Secret Store:**
    - **Meaning:** A secure repository for storing and retrieving sensitive information, often used for secrets management.
90. **Infrastructure Scaling:**
    - **Meaning:** Adjusting the capacity of computing resources based on demand to ensure optimal performance.
91. **Versioning:**
    - **Meaning:** Assigning unique identifiers or numbers to different versions of software or infrastructure configurations.
92. **Rollback:**
    - **Meaning:** Reverting to a previous version of an application or infrastructure configuration in case of issues with the latest update.
93. **Patch Management:**
    - **Meaning:** The process of updating and managing software patches to address security vulnerabilities and improve performance.
94. **Podcast:**
    - **Meaning:** A digital audio or video file available for streaming or downloading, often used for sharing insights and discussions in the tech community.
95. **Community of Practice:**
    - **Meaning:** A group of professionals with a shared interest or expertise who collaborate to learn and improve their skills.
96. **Continuous Integration (CI):** Automated process of integrating code changes into a shared repository multiple times a day.
97. **Continuous Deployment (CD):** Automated process of deploying code changes to production after passing CI tests.
98. **Pipeline:** An automated set of steps that code goes through from development to deployment.
99. **Infrastructure as Code (IaC):** Managing and provisioning infrastructure through code rather than physical hardware configuration.

100. **Docker:** Containerization platform for packaging, distributing, and running applications.
101. **Kubernetes:** Container orchestration platform for automating the deployment, scaling, and management of containerized applications.
102. **Microservices:** Architectural style that structures an application as a collection of small, independent services.
103. **Serverless:** Cloud computing model where cloud providers automatically manage the infrastructure, and developers focus on writing code.
104. **Scalability:** The ability of a system to handle increased load by adding resources or nodes.
105. **Monitoring:** Observing and tracking system performance, errors, and other metrics.
106. **Logging:** Recording events and activities within a system to aid in troubleshooting and analysis.
107. **Alerting:** Notification system that informs stakeholders about predefined events or issues.
108. **Version Control:** System for tracking and managing changes to source code.
109. **Git:** Distributed version control system widely used in software development.
110. **Repository:** Centralized location where version-controlled code is stored.
111. **Artifact:** A compiled or packaged piece of code or software.
112. **Binary Repository:** Storage for compiled binaries and artifacts.
113. **Jenkins:** Open-source automation server for building, testing, and deploying code.
114. **Ansible:** Automation tool for configuration management, application deployment, and task automation.
115. **Chef:** Configuration management tool for defining infrastructure as code.
116. **Puppet:** Configuration management tool for automating the provisioning and management of infrastructure.
117. **Terraform:** IaC tool for building, changing, and versioning infrastructure efficiently.
118. **Blue-Green Deployment:** A deployment strategy that reduces downtime by switching between two identical environments.
119. **Canary Release:** A deployment strategy that gradually rolls out a new version to a subset of users.
120. **Feature Toggle:** A technique to toggle features in an application on or off during runtime.

121. **Immutable Infrastructure:** An approach where once deployed, infrastructure components are never modified.
122. **Rollback:** Reverting a system to a previous state after a failed deployment.
123. **Zero Downtime:** A deployment strategy that ensures continuous availability during updates or changes.
124. **Dark Launch:** Introducing new features in a live environment but not exposing them to users.
125. **Failover:** The automatic switching to a backup system in case of a primary system failure.
126. **Load Balancer:** A device or service that distributes network traffic across multiple servers.
127. **Latency:** The time delay between the initiation of a request and the response.
128. **ChatOps:** Integrating chat tools into the DevOps workflow for collaboration and automation.
129. **Elasticity:** The ability of a system to automatically scale resources based on demand.
130. **Distributed System:** A system composed of multiple independent components that communicate and coordinate.
131. **Orchestration:** Coordinating and managing multiple automated tasks to achieve a specific outcome.
132. **Patch Management:** The process of keeping software and systems up-to-date with the latest patches.
133. **SLA (Service Level Agreement):** A formal commitment regarding the performance and availability of a service.
134. **SLO (Service Level Objective):** A target level of performance or reliability for a service.
135. **SLI (Service Level Indicator):** A measure of a specific aspect of a service's performance.
136. **Root Cause Analysis (RCA):** Investigating and identifying the primary cause of an incident or issue.
137. **Incident Response:** The process of managing and resolving incidents in a timely manner.
138. **CI/CD Pipeline:** A set of automated steps for continuous integration and continuous deployment.
139. **DevSecOps:** Integrating security practices into the DevOps process.
140. **Shift-Left Testing:** Performing testing earlier in the software development lifecycle.

141. **Infrastructure Monitoring:** Observing the performance and health of infrastructure components.
142. **Chaos Engineering:** Introducing controlled disruptions to a system to test its resilience.
143. **Automated Testing:** Using automation tools to execute tests and validate software.
144. **Test Driven Development (TDD):** Writing tests before writing the actual code.
145. **Container Registry:** A repository for storing and managing container images.
146. **Ephemeral:** Short-lived or temporary, often used to describe resources in cloud environments.
147. **Secret Management:** Securely storing and managing sensitive information such as passwords and API keys.
148. **Zero Trust Security Model:** A security model that assumes no trust and verifies each request.
149. **JWT (JSON Web Token):** A compact, URL-safe means of representing claims between two parties.
150. **OAuth:** An open standard for access delegation commonly used for authentication.
151. **Single Sign-On (SSO):** Allowing users to access multiple services with a single set of credentials.
152. **Capacity Planning:** Estimating the resources needed to support current and future workloads.
153. **Backlog:** A prioritized list of tasks or features yet to be addressed in a project.
154. **Burndown Chart:** A visual representation of completed and remaining work in a sprint or project.
155. **Agile:** A project management and product development approach that prioritizes flexibility and collaboration.
156. **Scrum:** An Agile framework for managing work with an emphasis on iterative and incremental development.
157. **Kanban:** A visual project management method for visualizing work, limiting work-in-progress, and maximizing flow.
158. **Velocity:** A metric in Agile development measuring the amount of work completed in a sprint.
159. **Story Points:** A measure used in Agile development to estimate the difficulty of implementing a user story.

160. **Retrospective:** A meeting held at the end of a sprint to review and improve the team's processes.
161. **Burnout:** A state of chronic physical and emotional exhaustion, often caused by prolonged stress.
162. **Pair Programming:** A development technique where two programmers work together at one workstation.
163. **Code Review:** A systematic examination of source code to find and fix errors.
164. **Technical Debt:** The cost of additional work required when code shortcuts are taken.
165. **Back-End:** The server-side of an application responsible for data processing and storage.
166. **Front-End:** The client-side of an application responsible for user interaction and presentation.
167. **Full Stack Developer:** A developer proficient in both front-end and back-end technologies.
168. **CICD Server:** A server responsible for managing and executing the CI/CD pipeline.
169. **Master Branch:** The main branch in a version control system where the source code is kept.
170. **Feature Branch:** A branch created to develop a new feature or enhancement.
171. **Artifact Repository:** A system for storing and managing artifacts produced by the CI/CD pipeline.
172. **Versioning:** Assigning unique identifiers to different versions of software or code.
173. **Rolling Deployment:** Gradual deployment of changes across a set of servers or instances.
174. **On-Premises:** Software or infrastructure hosted within an organization's physical location.
175. **Cloud Computing:** The delivery of computing services over the internet, often provided by third-party providers.
176. **Hybrid Cloud:** A combination of on-premises and cloud-based services.
177. **Public Cloud:** Cloud resources and services offered to the general public.
178. **Private Cloud:** Cloud resources and services dedicated to a single organization.
179. **Multi-Cloud:** The use of multiple cloud providers for different services or applications.

180. **Failover Cluster:** A group of servers that work together to maintain high availability.
181. **Bare Metal Server:** A physical server dedicated to a single customer, providing full control over hardware.
182. **Cold Start:** The initial startup of a serverless function.
183. **Warm Start:** The reuse of a pre-initialized serverless function.
184. **Hot Start:** The immediate execution of a serverless function without startup delay.
185. **Capacity Management:** Planning and managing the resources needed to meet demand.
186. **HAProxy:** An open-source load balancer and proxy server.
187. **Nginx:** A web server and reverse proxy server.
188. **SSL/TLS:** Protocols for securing data transmission over the internet.
189. **VPN (Virtual Private Network):** A secure network connection over the internet.
190. **CDN (Content Delivery Network):** A network of distributed servers to deliver web content.
191. **Distributed Tracing:** Monitoring and tracing the path of requests through a distributed system.
192. **Log Aggregation:** Collecting and centralizing log data from various sources.
193. **Compliance:** Adhering to legal and regulatory requirements in software development.
194. **Bastion Host:** A server that provides access to a private network from an external network.
195. **Dark Data:** Unused or unanalyzed data within an organization.