DevOps Certification Training

# Glossary of key terms

## DevOps Practices

1. DevOps: A culture and set of practices that aims to automate and integrate the processes between software development and IT operations teams.
2. Continuous Integration (CI): The practice of frequently integrating code changes into a shared repository, where automated builds and tests are run.
3. Continuous Deployment (CD): The practice of automatically deploying every code change that passes through the CI pipeline into production.
4. Agile: A software development methodology that emphasizes iterative development, collaboration, and customer feedback.
5. Scrum: An Agile framework for managing software development projects, involving iterative development cycles called sprints.
6. Kanban: A method for managing knowledge work with an emphasis on just-in-time delivery while not overloading the software development team members.
7. Version Control: The practice of tracking and managing changes to source code over time, typically using a version control system like Git.
8. Git: A distributed version control system used for tracking changes in source code during software development.
9. Repository: A central location where version-controlled files are stored, often referred to as a repo.
10. Branching: The practice of creating a parallel version of the codebase to work on a new feature or fix a bug without affecting the main codebase.
11. Merge: The process of combining changes from one branch (or multiple branches) into another, typically the main branch.
12. Pull Request (PR): A request to merge changes from one branch into another, often accompanied by a description of the changes and their purpose.
13. Build Automation: The process of automating the compilation, testing, and deployment of code changes.

## Containerization and Orchestration

1. Containerization: The practice of packaging an application and its dependencies into a standardized unit called a container for easier deployment and scalability.
2. Docker: A popular platform for containerization that allows developers to build, ship, and run applications in containers.
3. Orchestration: The automated arrangement, coordination, and management of computer systems, middleware, and services.
4. Kubernetes (K8s): An open-source platform for automating deployment, scaling, and managing containerized applications.

## Configuration Management Tools

1. Infrastructure as Code (IaC): The practice of managing and provisioning infrastructure through code and automation rather than manual processes.
2. Ansible: A configuration management tool that automates software provisioning, configuration management, and application deployment.
3. Chef: A configuration management tool used for managing infrastructure as code and automating the deployment of applications.
4. Puppet: A configuration management tool that automates the provisioning, configuration, and management of infrastructure.
5. SaltStack: A configuration management and automation tool that allows remote execution of commands and configuration tasks.
6. Terraform: An infrastructure as code tool that enables the building, changing, and versioning of infrastructure safely and efficiently.

## Incident and Change Management

1. Incident Management: The process of identifying, analyzing, and resolving incidents to restore normal service operation as quickly as possible.
2. Change Management: The process of controlling changes to the IT infrastructure to minimize disruption and ensure that all changes are authorized and documented.
3. Release Management: The process of planning, scheduling, and controlling the movement of changes to the IT infrastructure, including software releases and updates.
4. Configuration Management: The process of maintaining consistent and stable system configurations across different environments.
5. Capacity Planning: The process of determining the resources required to meet current and future demand.

## Monitoring and Reliability Engineering

1. Continuous Monitoring: The practice of monitoring systems and applications in real-time to detect and respond to issues quickly.
2. Log Management: The process of collecting, storing, and analyzing log data generated by systems and applications.
3. Metrics: Quantitative measurements used to assess the performance and health of systems and applications.
4. Service Level Agreement (SLA): A contract between a service provider and a customer that specifies the level of service expected.
5. High Availability (HA): A system design approach that ensures a high level of operational performance and uptime.
6. Disaster Recovery (DR): The process of recovering from a disaster or unexpected event that causes downtime or data loss.
7. Reliability Engineering: A discipline that focuses on ensuring that systems and applications meet reliability requirements under various conditions.

## Cloud Computing Models

1. Infrastructure as a Service (IaaS): A cloud computing model where virtualized computing resources are provided over the internet.
2. Platform as a Service (PaaS): A cloud computing model where a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure is provided.
3. Software as a Service (SaaS): A cloud computing model where software applications are hosted by a third-party provider and made available to customers over the internet.
4. Public Cloud: A type of cloud computing where resources are owned and operated by a third-party cloud service provider and delivered over the internet.
5. Private Cloud: A type of cloud computing where resources are used exclusively by one organization and can be hosted internally or externally.
6. Hybrid Cloud: A cloud computing environment that combines public and private clouds, allowing data and applications to be shared between them.
7. Multi-Cloud: A cloud computing strategy that involves using multiple cloud providers to avoid vendor lock-in and increase resilience.

## Cloud Computing Technologies and Concepts

1. Elasticity: The ability to scale computing resources up or down automatically based on demand.
2. Auto Scaling: The practice of automatically adjusting the number of compute resources in a cloud environment based on workload fluctuations.
3. Serverless Computing: A cloud computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers.
4. Microservices: An architectural style that structures an application as a collection of loosely coupled services.
5. API (Application Programming Interface): A set of rules and protocols that allows different software applications to communicate with each other.
6. Web Services: Software systems designed to support interoperable machine-to-machine interaction over a network.
7. RESTful API: An API that follows the principles of Representational State Transfer (REST), using standard HTTP methods like GET, POST, PUT, DELETE for communication.
8. SOAP (Simple Object Access Protocol): A protocol for exchanging structured information in the implementation of web services.
9. JSON (JavaScript Object Notation): A lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate.
10. XML (Extensible Markup Language): A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

## Security and Compliance

1. Identity and Access Management (IAM): The process of managing digital identities and controlling access to resources based on roles, permissions, and policies.
2. Network Security: The practice of securing a computer network infrastructure against unauthorized access, misuse, or modification.
3. Encryption: The process of encoding data to prevent unauthorized

IT Infrastructure

1. Virtualization: The process of creating a virtual representation of computing resources, such as servers, storage, or network devices.
2. Hypervisor: Software that enables multiple operating systems to share a single hardware host, allowing for virtualization.
3. Virtual Machine (VM): An emulation of a computer system that runs on virtualized hardware, allowing multiple VMs to run on a single physical machine.
4. Containerization: The practice of packaging an application and its dependencies into a standardized unit called a container for easier deployment and scalability.
5. Container Orchestration: The automated arrangement, coordination, and management of containerized applications, often using tools like Kubernetes.
6. Serverless Computing: A cloud computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers, allowing developers to focus on writing code.
7. Function as a Service (FaaS): A serverless computing model where developers write functions that are triggered by events, with the cloud provider managing the underlying infrastructure.
8. Infrastructure as a Service (IaaS): A cloud computing model where virtualized computing resources, such as servers, storage, and networking, are provided over the internet.
9. Platform as a Service (PaaS): A cloud computing model where a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure is provided.
10. Software as a Service (SaaS): A cloud computing model where software applications are hosted by a third-party provider and made available to customers over the internet.
11. Public Cloud: A type of cloud computing where resources are owned and operated by a third-party cloud service provider and delivered over the internet.
12. Private Cloud: A type of cloud computing where resources are used exclusively by one organization and can be hosted internally or externally.
13. Hybrid Cloud: A cloud computing environment that combines public and private clouds, allowing data and applications to be shared between them.
14. Multi-Cloud: A cloud computing strategy that involves using multiple cloud providers to avoid vendor lock-in and increase resilience.

## Software Engineering

1. Software Development Life Cycle (SDLC): The process of planning, creating, testing, and deploying software applications.
2. Agile: A software development methodology that emphasizes iterative development, collaboration, and customer feedback.
3. Scrum: An Agile framework for managing software development projects, involving iterative development cycles called sprints.
4. Kanban: A method for managing knowledge work with an emphasis on just-in-time delivery while not overloading the software development team members.
5. Version Control: The practice of tracking and managing changes to source code over time, typically using a version control system like Git.
6. Continuous Integration (CI): The practice of frequently integrating code changes into a shared repository, where automated builds and tests are run.
7. Continuous Deployment (CD): The practice of automatically deploying every code change that passes through the CI pipeline into production.
8. Code Review: The process of systematically examining another developer's code to ensure its quality, consistency, and adherence to coding standards.
9. Unit Testing: The practice of testing individual units or components of software to ensure they function correctly in isolation.
10. Integration Testing: The practice of testing the interfaces and interactions between software components to ensure they work together as expected.
11. Regression Testing: The practice of retesting software after changes to ensure that existing functionality has not been affected.
12. User Acceptance Testing (UAT): The process of testing software in a real-world environment by end-users to ensure it meets their requirements.
13. DevOps: A culture and set of practices that aims to automate and integrate the processes between software development and IT operations teams.
14. Microservices: An architectural style that structures an application as a collection of loosely coupled services.
15. API (Application Programming Interface): A set of rules and protocols that allows different software applications to communicate with each other.
16. Web Services: Software systems designed to support interoperable machine-to-machine interaction over a network.
17. RESTful API: An API that follows the principles of Representational State Transfer (REST), using standard HTTP methods like GET, POST, PUT, DELETE for communication.
18. SOAP (Simple Object Access Protocol): A protocol for exchanging structured information in the implementation of web services.
19. JSON (JavaScript Object Notation): A lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate.
20. XML (Extensible Markup Language): A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

## 

## Site Reliability Engineering (SRE) - Good to know (optional)

Site Reliability Engineering (SRE) is a discipline that originated at Google and focuses on ensuring the reliability, availability, and performance of large-scale, distributed software systems. SRE combines principles from software engineering and applies them to operations tasks to create scalable and reliable systems.

Key aspects of Site Reliability Engineering include:

1. Reliability: SRE aims to ensure that services are reliable and available, meeting user expectations for uptime and performance. This involves proactive monitoring, alerting, and incident response to minimize downtime and mitigate the impact of outages.
2. Automation: SRE emphasizes automation to reduce manual toil and improve operational efficiency. Automation is used for tasks such as provisioning infrastructure, deploying software updates, and responding to incidents.
3. Service Level Objectives (SLOs): SRE uses SLOs to define the level of service that a system should provide, based on user expectations and business requirements. SLOs help measure and manage the reliability of services and guide decision-making around resource allocation and trade-offs.
4. Error Budgets: SRE introduces the concept of error budgets, which represent the acceptable level of service degradation or downtime that a service can experience within a given time period. Error budgets help balance the need for reliability with the desire for innovation and rapid development.
5. Monitoring and Metrics: SRE relies on monitoring and metrics to measure the health and performance of systems and services. This includes collecting data on key indicators such as latency, error rates, and throughput, and using this information to make informed decisions about system improvements and optimizations.
6. Incident Management: SRE emphasizes rapid incident response and effective post-incident analysis to minimize downtime and prevent recurrence of issues. This involves establishing clear incident management processes, conducting blameless post-mortems, and implementing actionable improvements based on lessons learned.

Overall, Site Reliability Engineering aims to bridge the gap between development and operations teams by applying software engineering principles to operations tasks, with a focus on reliability, scalability, and automation. SRE practices have been adopted by many organizations beyond Google and are increasingly recognized as a valuable approach for building and managing reliable, scalable software systems.

### SRE Terms

1. Service Level Agreement (SLA): A contract between a service provider and a customer that specifies the level of service expected.
2. Service Level Objective (SLO): A target value for a service level indicator (SLI) that defines the desired level of reliability for a service.
3. Service Level Indicator (SLI): A quantifiable measure of the performance or quality of a service, often associated with SLAs and SLOs.
4. High Availability (HA): A system design approach that ensures a high level of operational performance and uptime, often achieved through redundancy and failover mechanisms.
5. Disaster Recovery (DR): The process of recovering from a disaster or unexpected event that causes downtime or data loss, often involving backup and replication strategies.
6. Incident Management: The process of identifying, analyzing, and resolving incidents to restore normal service operation as quickly as possible.
7. Problem Management: The process of identifying and resolving the root causes of incidents to prevent them from recurring.
8. Change Management: The process of controlling changes to the IT infrastructure to minimize disruption and ensure that all changes are authorized and documented.
9. Configuration Management: The process of maintaining consistent and stable system configurations across different environments.
10. Capacity Planning: The process of determining the resources required to meet current and future demand, often associated with IT infrastructure services.