50 DevOps Concepts that Cloud Engineers should know

1. **Continuous Integration (CI):**

   This practice requires developers to integrate code into a shared repository several times a day. Each integration is then verified by an automated build, allowing teams to detect problems early.

2. **Continuous Delivery (CD):**

   This is an extension of continuous integration, aiming at minimizing lead time, the time elapsed from development to actual production.

3. **Continuous Deployment:**

   This is a step further from Continuous Delivery where every change that passes all stages of the production pipeline is released to the customers. There's no explicit approval from a developer.

4. **Immutable Infrastructure:**

   The idea of Immutable Infrastructure is that once an instance is deployed, it never gets updated, it only gets replaced with a new instance.

5. **Microservices:**

   This architectural method is about developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

6. Infrastructure as Code (IaC):

This is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

7. Configuration Management:

This is about maintaining the system's configuration, keeping the system's information up-to-date, and easily accessible to the administrators.

8. Containerization:

Containers are a lightweight and standalone executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files.

9. Orchestration:

This involves managing the life cycles of containers, especially in large, dynamic environments.

10. Serverless Computing:

This is a cloud computing execution model where the cloud provider runs the server and dynamically manages the allocation of machine resources.

11. DevSecOps:

This involves integrating security practices within the DevOps process. DevSecOps involves creating a 'Security as Code' culture with ongoing, flexible collaboration between release engineers and security teams.

12. Monitoring and Logging:

Monitoring the health and availability of services, and logging for auditing and problem diagnosis purposes are essential practices in DevOps.

13. Performance Tuning:

It involves adjusting various system settings and components to achieve optimal performance.

14. Software-Defined Networking (SDN):

This is an approach to networking that separates the control and forwarding planes, and where control is directly programmable.

15. Blue/Green Deployment:

This technique reduces downtime and risk by running two identical production environments.

16. Canary Deployment:

A pattern for rolling out releases to a subset of users or servers. The idea is to test the waters and reduce the risk of any new release.

17. Rolling Deployment:

Here, updates are slowly rolled out across multiple instances, with new instances replacing old ones.

18. A/B Testing:

Also known as split testing, this is a way of comparing two versions of a deployment to see which one performs better.

19. Load Balancing:

This is the process of distributing network traffic across multiple servers to ensure no single server bears too much demand.

20. Git:

This is a version control system that makes collaboration easier, allowing changes to be tracked and providing an efficient way to handle merge conflicts.

21. Version Control Systems (VCS):

Software systems like Git that help multiple people manage changes to documents over time. VCS allows you to retrieve previous versions of your work and helps resolve conflicts when changes are made simultaneously by different people.

22. Code Review:

The systematic examination of software source code. It's done to find bugs, improve functions, or simply make code easier to understand.

23. Feature Flags:

A programming technique that provides an alternative to maintaining multiple source-code branches (known as feature branches), such that a

software feature can be tested even before it is completed and ready for release.

24. Test Automation:

The process of using special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes.

25. Unit Testing:

A level of software testing where individual components of a software are tested.

26. Integration Testing:

Testing phase in software testing where individual software modules are combined and tested as a group.

27. End-to-End Testing:

Ensures that the application works as expected and the flow of actions from start to finish performs without issues.

28. Smoke Testing:

Preliminary testing to check whether the implemented software build is stable or not.

29. Performance Testing:

Performed to determine how a system performs in terms of responsiveness and stability under a particular workload.

30. Cloud-Native Applications:

Applications that are built for the cloud, typically embodying microservices, containerization, and cloud orchestration.

31. Kanban:

A lean method to manage and improve work across human systems. This approach aims to balance demands with available capacity and improve the handling of system-level bottlenecks.

32. Scrum:

A subset of Agile, Scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.

33. Docker:

A popular open-source project that automates the deployment of software applications inside containers by providing an additional layer of abstraction and automation of OS-level virtualization on Linux.

34. Kubernetes:

An open-source platform designed to automate deploying, scaling, and operating application containers. It groups containers that make up an application into logical units for easy management and discovery.

35. Jenkins:

An open-source automation server that enables developers around the world to reliably build, test, and deploy their software.

36. Terraform:

An open-source Infrastructure as Code (IaC) software tool created by HashiCorp that enables users to define and provide data center infrastructure using a declarative configuration language.

37. Ansible:

An open-source software provisioning, configuration management, and application-deployment tool.

38. Puppet:

An open-source software configuration management tool which runs on many Unix-like systems as well as on Microsoft Windows, and includes its own declarative language to describe system configuration.

39. Chef:

A configuration management tool for dealing with machine setup on physical servers, virtual machines and in the cloud.

40. ELK Stack:

A collection of three open-source products — Elasticsearch, Logstash, and Kibana — all developed, managed and maintained by Elastic. ELK Stack is designed to allow users to take data from any source, in any format, and to search, analyze, and visualize that data in real time.

41. Continuous Integration (CI):

This DevOps practice involves regularly integrating code changes into a central repository, after which automated builds and tests are run.

42. Continuous Deployment (CD):

This is the practice of automatically deploying integrated changes to the production environment.

43. Continuous Delivery:

This extends continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

44. Infrastructure as Code (IaC):

This is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

45. Microservices:

A variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services.

46. Serverless Computing:

A cloud-computing execution model in which the cloud provider runs the server, and dynamically manages the allocation of machine resources.

47. Incident Management:

This practice involves identifying, analyzing, and correcting hazards to prevent a future recurrence.

48. SLI/SLO/SLA:

These are Service Level Indicators, Service Level Objectives, and Service Level Agreements, respectively, which are important metrics in DevOps.

49. ChatOps:

A collaboration model that connects people, tools, processes, and automation into a transparent workflow.

50. Observability:

This involves gathering different types of data about the functioning of a system so you can understand how it's working and why it behaves the way it does.