**DevOps**

**Cloud :** Cloud is a online storage space where people and business store their files and application, accessible from anywhere with the internet.

**Cloud Computing:** Cloud computing refers to the delivery of computing services such as storage, processing power, databases. Software and networking over the internet, rather than relying on local servers or personal devices.

Which have 2 modes

1. Service mode
2. Deployment mode

**Service mode or cloud computing service:**

1. **Infrastructure as a Service (IaaS) :**

* Provides the basic infrastructure – virtual machines, networking, storage – on a pay per – use basis.
* Examples : Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform.

1. **Platform as a Service (PaaS) :**

* Offers a platform allowing customers to develop, run, and manage applications without dealing with the underlying hardware and software infrastructure.
* Example: Google App Engine, Microsoft Azure App Service.

1. **Software as a Service (SAAS):**

* Deliver software applications over the internet, where users access the software through a web browser without installing or maintaining it on local devices.
* Example : Gmail, Microsoft Office, Salesforce, Dropbox.

1. **Function as a Service** **(FAAS)**:

* **Function as a Service**, a cloud computing model that allows developers to execute code in response to events without the need to manage the underlying infrastructure.
* Example : AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, IBM Cloud Functions.

**Types of Cloud Deployment Models:**

1. **Public Cloud**:
   * Services are offered over the public internet and are shared by multiple users. The cloud provider manages all infrastructure.
   * Examples: AWS, Microsoft Azure, Google Cloud.
2. **Private Cloud**:
   * Cloud infrastructure is used exclusively by a single organization. It may be hosted on-premises or by a third-party provider.
   * Suitable for businesses that require greater control and security.
3. **Hybrid Cloud**:
   * A combination of public and private cloud, allowing data and applications to be shared between them, providing greater flexibility and optimization of existing infrastructure.
4. **Community Cloud :**

* A community cloud is a type of cloud computing model that is shared by several organizations with common goals, interests, or regulatory requirements. Unlike public clouds, which are open to all users, and private clouds, which are dedicated to a single organization, a community cloud is designed to be shared by a specific community of users who have similar needs**.**

**Amazon Web Services (AWS) :**

* **Amazon Web Services (AWS)** is a comprehensive and widely adopted cloud platform offered by Amazon, providing a variety of cloud computing services. AWS delivers cloud-based solutions to individuals, businesses, and government organizations, helping them store data, host applications, and perform computing tasks without the need to maintain physical infrastructure.
* AWS starting in the year of 2005.
* Pay as you go means how much cloud use that much amount only we can pay.

**DevOps :** Combination of Developers team and Operations team.

* DevOps is the process of delivering the product /project/software by ensuring automation in place, ensuring the quality with continuous monitoring and continuous testing.

**Why DevOps …?**

* To deliver the Software / Project / Product on time.
* Improved Collaboration and Communication
* Higher Quality Software
* Greater Efficiency through Automation
* Faster Recovery and Reduced Downtime
* Cost Savings
* Scalability and Flexibility

**Software Development Life Cycle (SDLC) :**

The Software Development Life Cycle (SDLC) is a structured approach used by software developers and project managers to design, develop, test, and deploy software. The SDLC provides a systematic process for building software, ensuring it meets quality standards, business requirements, and user expectations.

**SDLC Models:**

**Waterfall Model:**

* Linear and Sequential Development model. And also non iterative model.
* A linear and sequential approach where each phase must be completed before moving to the next.
* Best suited for projects with well-defined requirements that are unlikely to change.
* Requirement analysis – System design – Implementation – Deployment – Maintenance.

**Requirement Analysis:**

* SRC document

**System Design :**

* Design – planning - assigning works/ tasks to the team.
* Coding will be done with Implementation.

**Implementation :**

* Integrate all the coding in previous step.
* Start testing.

**Deployment :**

* Complete software will be moved to global from local server.

**Maintenance :**

* Monitor Software.

**Agile Model:**

* An iterative and incremental approach where software is developed in small, manageable units (sprints), with frequent releases and ongoing feedback.
* Emphasizes flexibility, collaboration, and responding to change.
* Popular Agile methodologies include Scrum, Kanban, and Extreme Programming (XP).
* Requirement analysis – System Design – Development – Testing – Deployment – Maintenance / Support - Feedback – if Negative come it go to again to the System Design, It will happened till expected output come.

**Advantages :-**

* Requirement changes are allowed in any stage of development.
* Releases will be too fast. - > 1 week
* Customers no need to waiting for longer time.
* Good communication between all the teams.
* It is very easy to adopt.

**Disadvantages :**

* Less focus on design and documentation.

**Other models are :**

* **V – Model**
* **Spiral model**
* **Proto type**
* **Incremental model**

**Testing :**  3 types of testing

1. White box testing
2. Black box testing
3. Grey box

**1. White box testing:**

* Developers will test each and every line of the code.
* Need programming skills to design test cases.
* Developers fixes the bugs and perform 1 round of white box testing and send it to the testing team.

**2.Black box testing :**

* Examine the functionality of the software

There is a 2 types of testing

* 1. Functional testing
  2. Non Functional testing

**Functional testing :**

Functional testing refers to the testing of the functional aspects of an application, meaning the behaviour of the system or software based on specific inputs. This type of testing focuses on ensuring that the system behaves according to the functional requirements or specifications. The tester does not need to know how the system processes the inputs; they just need to verify that the outputs are correct based on the inputs.

**Examples of Functional Testing:**

* **Unit Testing:** Testing individual units or components of the software.
* **Integration Testing:** Testing the interaction between integratedcomponents or systems.
* **System Testing:** Testing the entire system to ensure all components work together.
* **Acceptance Testing:** Verifying if the system meets the business requirements and is ready for deployment**.**

**Non Functional testing :**

Non-functional testing, on the other hand, deals with non-functional aspects of the system, which are related to how well the system performs rather than what the system does. Non-functional testing does not focus on specific functional requirements, but rather on aspects like performance, security, usability, and scalability. This type of testing also does not require knowledge of the internal workings of the system.

**Examples of Non-Functional Testing:**

* **Performance Testing:** Testing the system's responsiveness, stability, and scalability under load (e.g., load testing, stress testing).
* **Usability Testing:** Assessing the user interface and overall user experience (UX).
* **Security Testing:** Testing how secure the system is from vulnerabilities and external threats.
* **Compatibility Testing:** Verifying that the application works on different devices, browsers, or operating systems.
* **Reliability Testing:** Ensuring that the system can perform consistently over time.

**Bugs :** A bug is a general term used to describe a flaw, fault, or unintended behaviour in a software system that leads to incorrect or unexpected results. Bugs are typically caused by mistakes made during the software development process (e.g., coding errors, miscommunication, or incorrect assumptions).

**Error / Mistake :** An error refers to a mistake or issue made by a developer, which can lead to incorrect code or unintended behaviour in the system. It is a human-made issue in the software, often during coding or design**.**

**Defect :** A defect is a flaw or imperfection in the software that causes it to deviate from the expected behaviour or functional requirements. A defect typically emerges from an error in the code or design phase.

**Failure :** failure in software is when the software doesn't work as expected or doesn't do what it's supposed to do when it's used**.**

**Top 10 cloud providers :**

AWS – 36%

AZURE – 27%

GCP – 15%

ALIBABA – 10%

TENCENT – 7%

ORACLE – 3%

IBM – 7%

DIGITAL OCEAN

SALESFORCE - 11%

VIMVARE

**Tools required :-**

* 1. **Planning / Coding / SCM :** GIT, JIRA
  2. **Building code :** Maven, Gradel, Apache ANT
  3. **Testing :** Selenium testing
  4. **Integration :** Jenkins
  5. **Deployment :** Docker, Kubernetes
  6. **Operations :** Ansible
  7. **Monitoring :** Terraform