**Software Development Life Cycle (SDLC)**

SDLC is a process for developing and maintaining software systems. It consists of the following phases:

* **Requirements analysis**: understanding and documenting user requirements.
* **Design**: creating a high-level system design and detailed technical specifications.
* **Implementation**: writing code according to the design specifications.
* **Testing**: verifying that the software meets requirements.
* **Deployment**: installing and configuring the software in the production environment.
* **Maintenance**: ongoing support, updates, and bug fixes. Common SDLC models include Waterfall, Agile, Scrum, DevOps, and RAD. Best practices include documentation, quality assurance, communication, version control, security, and continuous improvement.

Before DevOps, there were two development models:

1. **Waterfall model**

Waterfall model is a sequential software development process model that follows a linear and sequential approach.

In this model each phase of software development process must be completed before the next phase can begin. The requirements are defined and documented at the beginning of the project, and any changes to the requirements are not permitted once the project has begun.

**Disadvantages:**

* It is difficult to estimate time and cost for each phase of the development process.
* Once an application is in testing stage, it is very difficult to go back and change something that was not well thought out in the concept stage.
* Not a good model for complex and object-oriented projects.
* Not suitable for projects where requirements are at a moderate to high risk of changing.
* It does not encourage the clients to give new requirements. Not allow us to go back.

**Agile model:**

Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

* Planning
* Requirements Analysis
* Design
* Coding
* Unit Testing and
* Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

* Agile development model is also a type of incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained.
* Agile software development method emphasis on iterative, incremental and evolutionary development.

Waterfall model – 1 Year

Agile model – 6months --> 50% speeds up

**Delay:**

* In spite of the model, still delay
* Gap between development team and operations team.

**Operations team responsibilities:**

* Installation of server hardware and operating system.
* Configuration of servers, networks, storage, etc.
* Monitoring of servers
* Respond to outages
* IT security
* Change control
* Backup and disaster recovery planning

**DevOps and its importance:**

DevOps is a methodology that promotes collaboration between development and Operations team. This allows deploying code to production faster and in an automated way. It helps to enables rapid deployment of products.

**Why we need DevOps?**

* Faster delivery
* Reduce the complexity in traditional IT
* To save time
* Quality product – fully tests
* Cost saving – less expensive as compared to traditional IT
* Better resource management
* Continuous integration – automation
* Stability and reliability

**DevOps prerequisites:**

1. Linux Fundamentals
2. Source Code Management (SCM) – Git
3. Build tool – Maven (To build an artifact)
4. Continuous Integration and Continuous delivery (CI/CD is the heart of devops) – Jenkins
5. Containerization – Docker (all about provisioning the infrastructure)
6. Container Orchestration - Kubernetes
7. Configuration Management tool -Ansible(Infrastructure as Code)
8. Monitoring tool – Nagios
9. Continuous Testing – Selenium
10. Cloud service knowledge – AWS, Azure, GCP

**DevOps Lifecycle:**

1. **Continuous Development:**

* Involves ‘planning’ and ‘coding’ of the software.
* Version control tools maintain code
* Git, SVN, Mercurial, CVS, etc. are some popular examples.

1. **Continuous Testing:**

* Developed software is continuously tested for bugs.
* Automation tools like selenium, TestNG, Junit etc. are used.
* Automations saves times and efforts.

1. **Continuous Integration:**

* Heart of the entire DevOps lifecycle.
* Developers require to commit changes to the source code more frequently
* Code supporting new functionality is continuously built with the existing code.

1. **Continuous Deployment:**

* Code is deployed to production servers.
* Configuration management is an important practice.
* Some popular tools that are used here are puppet, chef, saltstack and ansible.

1. **Continuous Monitoring**

* Continuously monitor the performance of your application.
* System errors are resolved in this phase.
* The popular tools used for this are Splunk, ELK Stack, Nagios, NewRelic and Sensu.

**Azure and AWS DevOps**

* 1. **Azure DevOps:**
* Set of collaborative development tools built for the cloud.
* Azure DevOps servers.
* Azure DevOps services.
  1. **AWS DevOps:**
* AWS is one of the best cloud service providers in the market today.
* They are many reasons as to why AWS DevOps is a highly popular combination.