**DevOps:**

**Def:** DevOps is a software development approach which involves,

1) Continuous Development

2) Continuous Testing

3) Continuous Integration

4) Continuous Deployment

5) Continuous Monitoring

of the software throughout the development lifecycle.

The intention of DevOps is to create better-quality software more quickly and with more reliability while causing greater communication and collaboration between teams. This is the key to understanding what is DevOps!

**Tools:** Tools that are used to automate these stages are:

1. Git, SVN : Source Code Management
2. Maven, Ant, Gradle : Building
3. Selenium, Junit : Testing
4. Jenkins : Integration
5. Puppet, Chef, Saltstack, Ansible : Deployment
6. Nagios, Zenoss, etc.,, : Monitoring

DevOps integrates developers and operations team to improve collaboration and productivity.

**Collaboration** : Is a plan that matches short-term and long-term goals with specific technology solutions to help meet those goals. It is a plan that applies to a new product or process, or to an emerging technology.

**Productivity** : The ratio between the functional value of software produced to the labor and expense of producing it.

According to the DevOps culture, a single group of Engineers (developers, system admins, QA’s, Testers etc. turned into DevOps Engineers) has end to end responsibility of the Application (Software) right from gathering the requirement to development, to testing, to infrastructure deployment, to application deployment and finally monitoring & gathering feedback from the end users, then again implementing the changes.

**Dev Challenges :**

1. Waiting time for code deployment
2. Pressure of work on old, pending and new code

**DevOps Solution** :

1. Continuous Integration ensures that there is quick deployment of code, faster testing and speedy feedback mechanism.
2. There would be no waiting time to deploy the code. Hence, the developers focuses on building the current code.

**Ops Challenges** :

1. Difficult to maintain uptime of the production environment
2. Tools to automate the infrastructure management are not effective
3. No. of servers to be monitored increases
4. Difficult to diagnose and provide feedback on the product

**DevOps Solution** :

1. **Containerization / Virtualization** ensures that there is a simulated environment created to run the software as containers offer great reliability for service uptime.
2. **Configuration Management** helps to organize and execute configuration plans, consistently provision the system, and proactively manage their infrastructure.
3. **Continuous Monitoring** ensures effective monitoring and feedback s systems and is established through powerful monitoring tools like Nagios, Zenoss, etc.

**Evolution of Software Development**

DevOps evolved from existing software development strategies/methodologies over the years in response to business needs. Let us briefly look at how these models evolved and in which scenarios they would work best.

1. **Traditional Waterfall Model**

* Complete requirements are clear and fixed
* Product definition is stable

1. **Agile Development Model**

* Requirements change frequently
* Development needs to be fast

1. **DevOps Approach**

* Requirements change frequently
* Development needs to be Agile
* Operations needs to be Agile