What is DevOps?

A concept that has emerged because of the failures of traditional software release methods (Waterfall).

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

This speed enables organizations to better serve their customers and compete more effectively in the market.

DevOps = Development + Operations

=> DevOps is a culture

=> DevOps is a process

=> DevOps is set of practices

=> DevOps is used to establish collaboration between Development team & Operation team.

Note: The main aim of DevOps is to delivery application to client quickly with high quality

=> As a DevOps engineer, we will automate application build and deployment process

using DevOps tool chain.

=> DevOps practices will help us in Simplifying and Automating Application Build & Deployment Process.

Software Development Life Cycle (SDLC)

Typical Release process

****

1. Idea to requirements gathered by Business Analysts
2. Software developers take the technical requirements and code them.
3. Testers will write test cases and test the software
4. Systems administrators prepare and configure servers/databases (Includes security, firewalls, installation of dependencies, etc....)
5. Deploy servers for users to use. (Build, package and deploy.)
6. Operations and monitoring.
   1. Are there bugs that were not caught during dev/testing?
   2. How is the system responding to high user load?

3. Are there any kind of attacks from malicious actors?

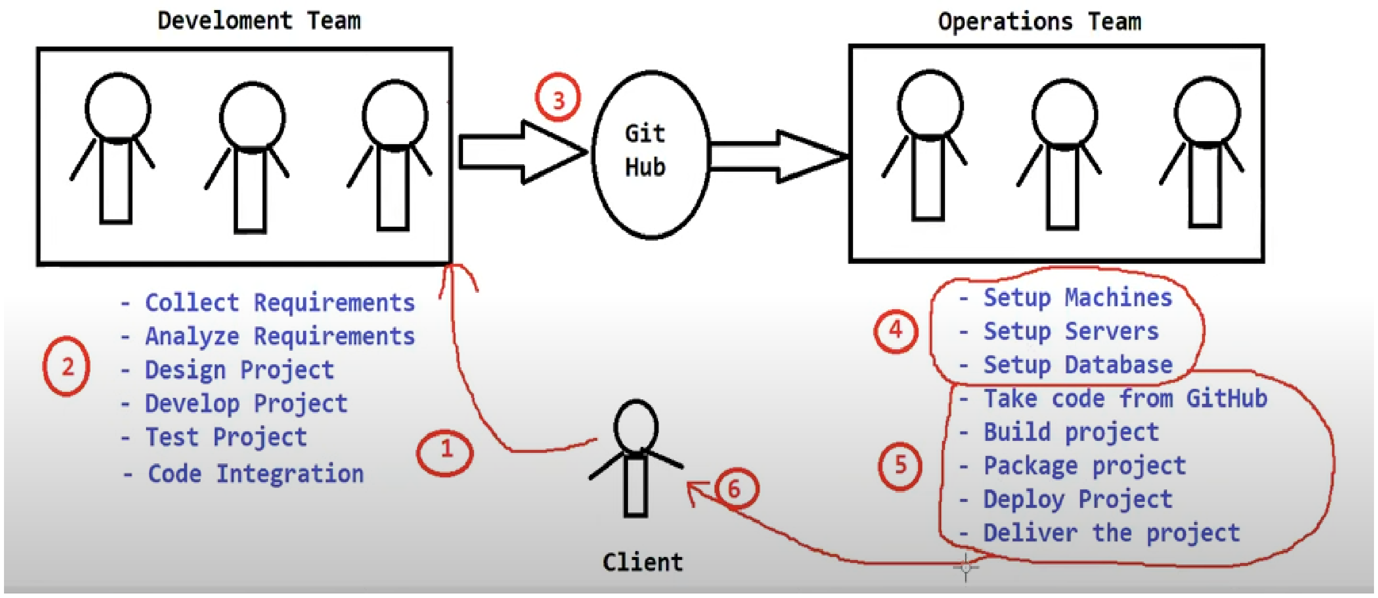
4. Are the disks filling up?

5. Is the CPU/Memory under or overutilized?

6. Are there new vulnerabilities caused by dependent libraries used in the application?

7. Are there parts of the system throwing errors that are not visible to the frontend?

7. Improvements to the software - new releases...new updates..new versions (1.0.0)..start the cycle again (1.0.1).....(1.1.4).....(2.0.0)



Problems with the traditional approach (Waterfall)

****

1. Miscommunication and lack of comms between developers and operations.

2. Usually, there will be produced documentation on release steps and checklists on what must be done to achieve a successful release. There is NO automated process.

3. Release takes much longer due to miscommunication

4. Conflicting of interests

5. Communication problems between operations and the Security Team

6. End to end Testing problems which is done by a different team

7. All the entire steps to release is done manually - Slow and prone to human errors

Agile methodology

****

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

Scrum:

Scrum is an agile project management framework that helps teams structure and manage their work through a set of values, principles, and practices.

Sprint:

A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work.

Kanban:

Kanban is a popular framework used to implement agile and DevOps software development. It requires real-time communication of capacity and full transparency of work. Work items are represented visually on a kanban board, allowing team members to see the state of every piece of work at any time.

DevOps to the rescue - Quickly deliver HIGH-QUALITY code by removing all the blockers that cause a slow down to the release process.

1. Fast - new patch releases can get to users in minutes, major releases can get to users in hours/few days

2. Improved communication between developers and operators

3. Automated process - every step of the cycle is automated.

More releases can be done in small increments, rather than wait till the entire software is completed.

4. Quality code - with automation comes to a lot of great benefits.

Since code is automatically tested, and badly written code cannot make it to end-users, then we have a solid product

5. More secured application/infrastructure since DevOps also strongly considers security in the process - DevSecOps

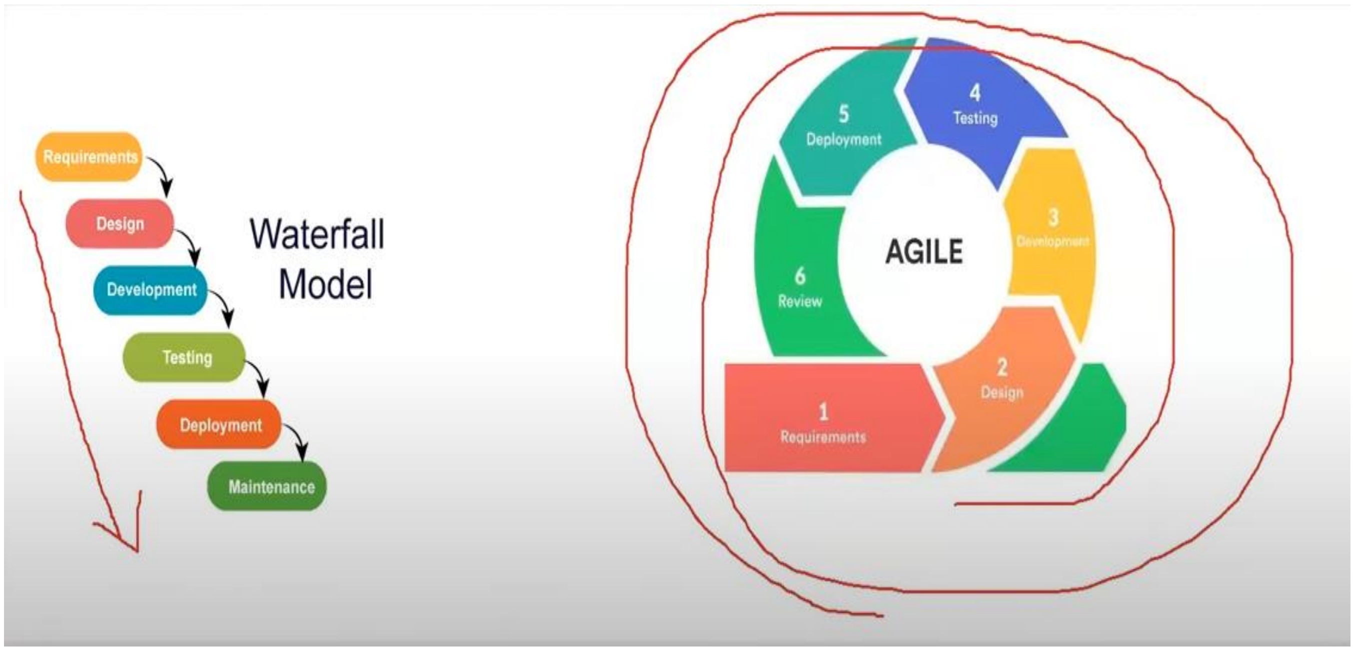
6. Because both apps and infrastructure is automated, it is easy to replicate the entire stack to the exact state it was before it got destroyed,

and the software can be back in business in just minutes. You don't have to remember any documentation or follow checklists in whatever order.

Simply executing DevOps code solves this problem.

7. Much happier customers.

8. Much quicker feedback for further improvements



==

DevOps Tools: To Automate Operations required for the project

==

1) **Maven / Gradle**: Build Tools (Compile + Unit Test Execute + Package as jar / war)

2) **Git Hub / BitBucket**: Source Code Repository Servers (To integrate source code)

3) **SonarQube** : Code Review Software (To identify developers mistakes in code)

4) **Nexus / JFrog** : Artifactory Repository Servers (To store build artifacts)

5) **Apache Tomcat** : WebServer (To execute our java web application)

6) **Jenkins** : Deployment Tool (CI CD)

7) **Terraform / CloudFormation** : Insfrastructure creation tool (AWS / Azure / GCP)

8) **Ansible / Chef / Puppet** : Configuration Management Software

9) **Docker** : Containerization Platform (Build + Ship + Run-Anywhere)

10) **Kubernetes (K8S) / SWAM** : Orchestration Platform (To manage docker containers)

11) Monitoring Tools

- Grafana

- Prometheus

- **ELK Stack (Elastic Search + Log Stash + Kibana)**

- Splunk

- Newrelic

=========================

Roles Of Development Team

=========================

1) Understand Requirements

2) Analyze Requirements

3) Designing

4) Development / Coding

5) Unit Testing

6) Code Integrating (Source Code Repository Server)

================================

Role of QA / Testing Team (Quality Assurance Team)

================================

1) Prepare Test Scenarios

2) Prepare Test Cases

3) Execute Test cases on the project

4) Report Bugs

5) Provide QA Approval

========================

Roles of Operations Team

========================

1) Setup Infrastructure (terraform / CloudFormation)

- Machines / Servers

- Databases

- Storage

- Network

- Backup

2) Configuration management (Ansible)

3) Source Code Repository Management (Git Hub / BitBucket / SVN / Codecommit)

4) Build the Source Code (Maven / Gradle)

=> Take the latest source code from the repository

=> Compile the source code

=> Execute Unit Test cases

=> Package the code as jar / war file

5) Perform Code Review and Generate Code Review Report (SonarQube)

6) Store Build Artifact into Artifactory Repository Server (Nexus / JFrog)

=> Storing jar / war file for backup)

7) Deploy Project Artifact into Server (Apache Tomcat / JBoss / IIS)

=> Keeping the project war file in server is called Deployment

8) Automate Build and Deployment process for quick deliveries (Jenkins / Bamboos / UDeploy)

=> For several environments:

- DEV

- QA

- UAT

- Pilot (PRE-PROD)

- PROD

9) Execute the application in containers - Containerization (Docker)

Note: Using Docker we can easily execute our application in any platform without bothering about underlying infrastructure.

10) Orchestrate Docker Containers - kubernetes (K8S) / Openshift

Note: Orchestration means Management. K8S will manage Docker containers

(Create /scale up /scale down / destroy)

11) Monitor Infrastructure and Application

- Grafana

- Prometheus

- ELK Stack (Elastic Search + Log Stash + Kibana)

- Splunk

- Newrelic

- Nagios