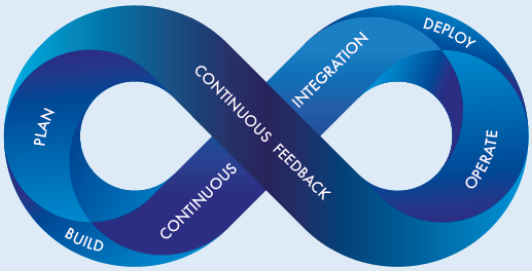


Dev-Ops Fundamentals

An Introduction guide

Riyaz Sheik

Contents

INTRODUCTION

SDLC BASICS

SDLC stands for Software Development Life Cycle. SDLC is a process that consists of a series of planned activities to develop or alter the Software Products.

Phase 1: Planning

In the planning phase, project goals are determined and a high-level plan for the intended project is established. Planning is the most fundamental and critical organizational phase. The three primary activities involved in the planning phase are as follows:

- Identification of the system for development
- Feasibility assessment
- Creation of project plan

Phase 2: Analysis

In the analysis phase, end user business requirements are analyzed and project goals converted into the defined system functions that the organization intends to develop. The three primary activities involved in the analysis phase are as follows:

- Gathering business requirement
- Creating process diagrams
- Performing a detailed analysis

Business requirement gathering is the most crucial part at this level of SDLC. Business requirements are a brief set of business functionalities that the system needs to meet in order to

be successful. Technical details such as the types of technology used in the implementation of the system need not be defined in this phase. A sample business requirement might look like “The system must track all the employees by their respective department, region, and the designation”. This requirement is showing no such detail as to how the system is going to implement this requirement, but rather what the system must do with respect to the business.

Phase 3: Design

In the design phase, we describe the desired features and operations of the system. This phase includes business rules, pseudo-code, screen layouts, and other necessary documentation. The two primary activities involved in the design phase are as follows:

- Designing of IT infrastructure
- Designing of system model

To avoid any crash, malfunction, or lack of performance, the IT infrastructure should have solid foundations. In this phase, the specialist recommends the kinds of clients and servers needed on a cost and time basis, and technical feasibility of the system. Also, in this phase, the organization creates interfaces for user interaction. Other than that, data models and entity relationship diagrams (ERDs) are also created in the same phase.

Phase 4: Implementation

In the development phase, all the documents from the previous phase are transformed into the actual system. The two primary activities involved in the development phase are as follows:

- Development of IT infrastructure
- Development of database and code

In the design phase, only the blueprint of the IT infrastructure is provided, whereas in this phase the organization actually purchases and installs the respective software and hardware in order to support the IT infrastructure. Following this, the creation of the database and actual code can begin to complete the system on the basis of given specifications.

Phase 5: Testing & Integration

In the testing phase, all the pieces of code are integrated and deployed in the testing environment. Testers then follow Software Testing Life Cycle activities to check the system for errors, bugs, and defects to verify the system’s functionalities work as expected or not, often. The two primary activities involved in the testing phase are as follows:

- Writing test cases
- Execution of test cases

Testing is a critical part of software development life cycle. To provide quality software, an organization must perform testing in a systematic way. Once test cases are written, the tester executes them and compares the expected result with an actual result in order to verify the system and ensure it operates correctly. Writing test cases and executing them manually is an intensive task for any organization, which can result in the success of any business if executed properly.

During the Integration phase, the system is deployed to a real-life (the client's) environment where the actual user begins to operate the system. All data and components are then placed in the production environment. This phase is also called referred to as 'delivery.'

Phase 6: Maintenance

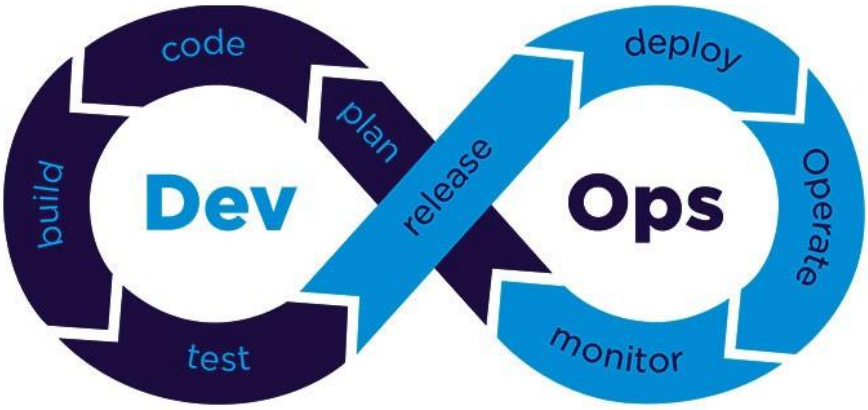
In the maintenance phase, any necessary enhancements, corrections, and changes will be made to make sure the system continues to work, and stay updated to meet the business goals. It is necessary to maintain and upgrade the system from time to time so it can adapt to future needs. The three primary activities involved in the maintenance phase are as follows:

- Support the system users
- System maintenance
- System changes and adjustment

BASICS OF DEV-OPS

DevOps is a software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops). It strongly advocates automation and monitoring at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management. DevOps aims at shorter development cycles, increased deployment frequency, and more dependable releases, in close alignment with business objectives. DevOps benefits allow to deliver products at a higher speed, allowing to easily scale and in a more reliable way.

DevOps is a software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops). It strongly advocates automation and monitoring at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management. DevOps aims at shorter development cycles, increased deployment frequency, and more dependable releases, in close alignment with business objectives. DevOps benefits allow to deliver products at a higher speed, allowing to easily scale and in a more reliable way.



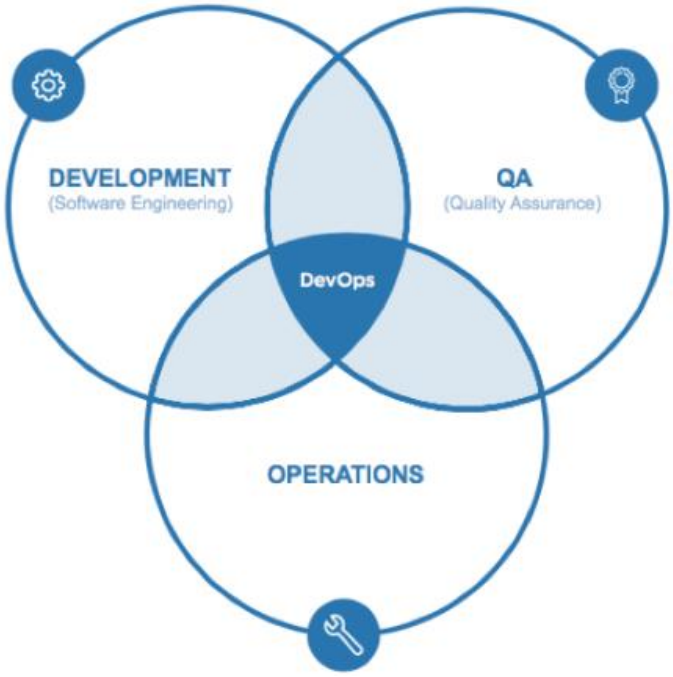
This revolutionary way of managing software it is based on two key concepts:

- Cross Functional Teams: Rapid collaboration is essential, there is no separation between development, operations, database, and testing teams. Organization will have a single and coordinated IT department. Each team must have all the skills to build, ship and maintain their software.
- Automation: Container based technologies aligned with the micro service architecture have changed the way software is built allowing full automation of all the software development cycles including development, testing, deployment and monitoring; even the IT infrastructure can be fully automated. It is important to note that big monolithic apps and large microservices are both complex systems and micro services doesn't simplify the deployment, it may even complicate more. What DevOps brings compared to monolithic apps if full automation thanks to tools like Ansible.

At the heart of DevOps are the Three Ways:

- The principles of Flow, which accelerate the delivery of work from Development, to Operations, to our customers.
- The principles of Feedback, which enable us to create ever safer systems of work.
- The principles of Continual Learning and experimentation, which foster a high-trust culture and a scientific approach to organizational improvement as part of our daily work.

These 3 ways are core guidelines and best practices for agile development and exploration. Following these principals is key for a business transformation.



The way teams develop software is constantly evolving. A decade ago, most software development environments were siloed, consisting of software developers in one silo and mainframe computers and a staff of IT professionals who maintained that mainframe in another silo. Today, virtualization allows us to create and proliferate virtual machines almost instantly. These technologies enable us to automate more tasks and command larger, more powerful infrastructures to increase the efficiency of our software development operations. We are now in a technological revolution and DevOps it is the core principle of this revolution.

Best Practices

#1. Test Automation

To compose quality code, developers need to test the software regularly. DevOps allows for early testing that gives developers an opportunity to identify and resolve the issue during software development rather than later in the process.

Automated testing allows for quicker execution of SDLC (Software Development Life Cycle) in comparison to manual testing. Test automation can be applied to the database and networking changes, middleware configurations, and code development using regression testing and load testing.

Test automation can be accomplished by performing varied activities such as identifying test scenarios and cases, choosing the right set of tools for automation, setting up an appropriate test environment, running test cases and analyzing results.

#2. Integrated Configuration Management

Configuration and change management are integral parts of the operations. Configuration management is about automation, monitoring, management, and maintenance of system-wide configurations that take place across networks, servers, application, storage, and other managed services.
Integrated configuration management enables the development teams with a bigger picture. It allows to utilize the existing services during the software development rather than investing time and efforts in reinventing the new services from scratch.

#3. Integrated Change Management
Change management is a process in which configurations are changed and redefined to meet the conditions of dynamic circumstances and new requirements. During the configuration management, if any changes are required then change management comes into the picture. The operations teams provide their inputs on what opportunities and consequences the change might expose at the wider level and what other systems could be impacted.

#4. Continuous Integration
It is a DevOps software development practice where the development team regularly updates the code changes in the repository after which the automated builds and tests run. Continuous Integration practices allow developers to carry out integrations sooner and frequently. Whereas CI tools help them to detect the integration challenges in new and existing code and solve them at the earlier phase only. Thus, CI improves collaborations amongst the teams and ultimately builds a high-quality software product.

#5. Continuous Delivery
Continuous Delivery is a DevOps practice where the newly developed code is updated by developers, get tested by the QA team at the different stages by applying both automated and manual testing, and once the case passes all the testing, it gets into the production. It allows the development team to build, test and release the application faster and frequently, in short cycles. It helps organizations to increase the number of deliveries, reduce manual works, minimize the risk of failure during production, and more.

#6. Continuous Deployment
The deployment process contains varied sub-processes such as code creation, testing, versioning, deployment, post-deployment, etc. In the continuous deployment process, the code is automatically deployed in the production environment once it successfully passes all the test cases in QA, UAT, and other environments. A lot of tools available that perform continuous deployment start from staging to production with minimum human intervention.

#7. Application Monitoring
App infrastructure monitoring is very crucial to optimize the application performance, whether it is deployed on the cloud or local data center. If a bug hits the application during the release process, then it will be turned into the failure. So, it is very important for the development teams and operations teams to consider proactive monitoring and check the performance of the

application. Various tools are available for application monitoring that offer a lot of metrics related to applications, infrastructure, sales, graphs, analytics, etc.

#8. Automated Dashboards

Leverage the DevOps intelligence with the automated dashboard. It will provide the data along with detailed insights and reports of every operation such as the number of tests run, tests' durations, the number of failure and success in testing. It allows to review configuration changes made to the database and server and deployments that have taken place across the system.

The dashboard acts as a centralized hub that enables the operations team with real-time data insights which help them in selecting the right set of automation tools testing. Moreover, there are varied logs, graphs, and metrics that enable operations teams with a holistic view of changes happening in the system.

Agile and Dev-ops

Generic Use cases

DEV-OPS CONCEPTS

JENKINS, PIPELINES

AUTOMATION OF BUILDS

CONTINUOUS INTEGRATION

CONTINUOUS DEPLOYMENT/DELIVERY

CONFIGURATION MANAGEMENTS

Advanced concepts in Dev-Ops

Dev-ops on Cloud - AWS & Azure

Infrastructure as Code

Virtualization & Containerization

Orchestrations

Monitoring