Top 12 DevOps Tools for Your DevOps Implementation Plan

Here are the DevOps tools you need to keep under your belt.

Mitul Makadia, Dec. 11, 19
(https://dzone.com/articles/top-12-devops-tools-for-your-devops-implementation?edition=553297&utm_source=Daily%20Digest&utm_medium=email&utm_campaign=Daily%20Digest%202019-12-13)

DevOps is a software development and delivery process that helps in emphasizing communication along with cross-functional collaboration between product management, software development, and operations professionals.

We've curated a list of the Top 12 DevOps Tools, along with their features, based on our decade-long experience in the IT industry, dealing with infrastructure for the significant part. We've taken great care in selecting, benchmarking, and continuously improving our tool selection. That apart, the article also covers the DevOps transformational roadmap as well as the step by step implementation guide.

The popularity of DevOps, in recent years, as robust software development and delivery process has been unprecedented. As we talked about in our previous piece of the same series, DevOps is essentially the integration of two of the essential verticals in IT – development and operations – that brings a whole new perspective to the execution of software development. DevOps implementation is mainly about bringing a cultural transformation where both development and operations teams collaborate and work seamlessly. Let us learn about DevOps implementation strategy and the top DevOps tools available in the market today.

You may also enjoy: Most Popular DevOps Tools You Must Learn In 2019

The primary goal of DevOps is to improve collaboration between various stakeholders, from planning and deployment to the maintenance of the IT project, to be able to:

- Improve the frequency of deployment
- Reduce the time between updates/fixes
- Achieve speedy delivery
- Improve time to recovery
- Reduce the failure rate of new releases

DevOps Transformational Roadmap

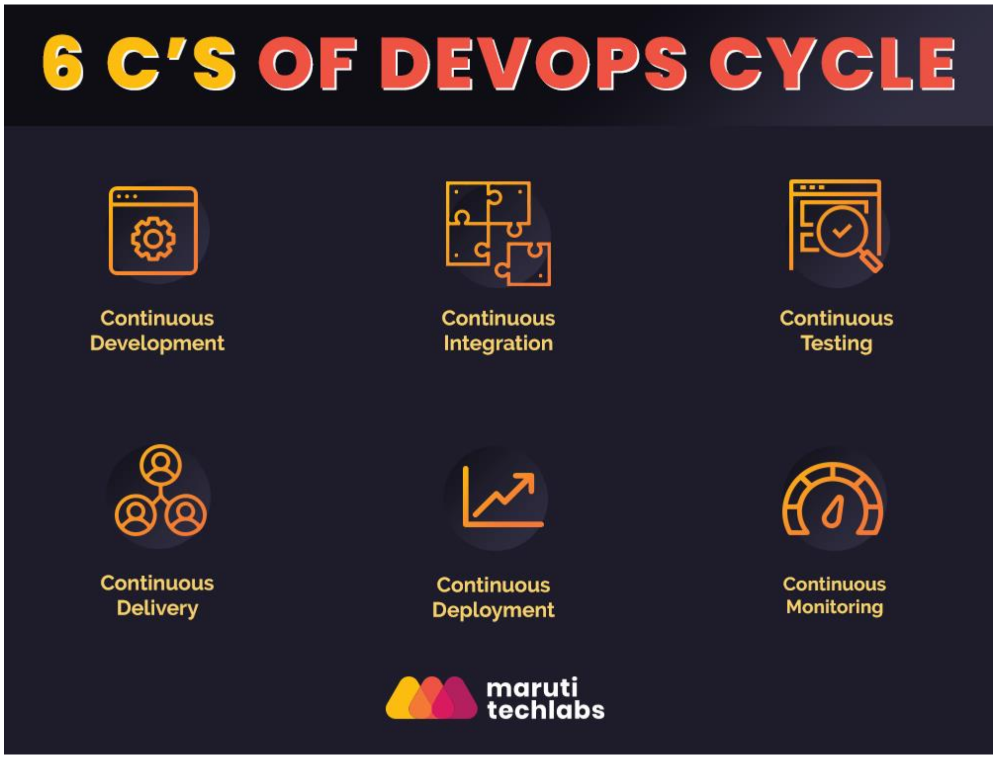
The DevOps implementation approach is categorized into three main stages of the software development lifecycle:

- Build (DevOps Continuous Integration)
- Test (DevOps Continuous Testing)

- Release (DevOps Continuous Delivery)

The concept of DevOps implementation integrates development, operations, and testing departments into collaborative cross-functional teams with the aim of improving the agility of overall IT service delivery.

The focus of DevOps is mostly on easing delivery processes and standardizing development environments with the aim of improving efficiency, security, and delivery predictability. DevOps empowers teams and gives them the autonomy to build, deliver, validate, and support their software applications. It provides developers with a better understanding of the production infrastructure and more control of the overall production environment.



DevOps cycle

As an organization, your DevOps journey begins by defining the existing business procedures, IT infrastructure, and delivery pipelines, followed by crafting clear objectives that the DevOps implementation strategy is expected to achieve for your organization.

Although DevOps is implemented with different variations in different organizations, the common phases of the DevOps process consist of the 6C's as discussed below:

- **Continuous Developmen**t – Continuous development involves planning, outlining, and introducing new code. The aim of continuous development is to optimize the procedure of code-building and to reduce the time between development and deployment.
- **Continuous Integration (CI)** – This practice of DevOps implementation involves the integration of developed code into a central repository where configuration management (CM) tools are integrated with test & development tools to track the code development status. CI also includes

quick feedback between testing and development to be able to identify and resolve various code issues that might arise during the process.

- **Continuous Testing** – The aim of continuous testing is to speed up the delivery of code to production. This phase of DevOps involves simultaneous running of pre-scheduled and automated code tests as application code is being updated.
- **Continuous Delivery** – Continuous delivery is aimed at quick and sustainable delivery of updates and changes ready to be deployed in the production environment. Continuous delivery ensures that even with frequent changes by developers, code is always in the deployable state.
- **Continuous Deployment (CD)** – This practice also automates the release of new or changed code into production, similar to continuous delivery. The use of various container technology tools such as Docker and Kubernetes allow continuous deployment as they play a key role in maintaining code consistency across different deployment environments.
- **Continuous Monitoring** – It involves ongoing monitoring of the operational code and the underlying infrastructure supporting it. Changes/application deployed in the production environment is continuously monitored to ensure stability and best performance of the application.

Advantages of DevOps

Some of the key benefits of DevOps implementation include:

- Speedy and better product delivery
- Scalability and greater automation
- High clarity into system outcomes
- Stable operating environments
- Better utilization of resources
- High clarity into system outcomes

Does that mean there are no hurdles to DevOps adoption?

Not necessarily! Similar to any other approach, DevOps adoption also comes with certain hiccups. Although the concept of DevOps is a decade old now, there are certain aspects that need to be taken care of so that they don’t become hurdles in embracing the collaborative IT practice. Let us have a look at some of the key points.

a) Cost Reduction

DevOps implementation reduces the number of project failures and rollbacks, and as a result, reduces the overall IT cost in the long run. However, if not planned properly, the cost of shifting to DevOps practice can burn a hole in your pocket. Planning the budget is a crucial step before DevOps implementation.

b) Skill Deficiency

Hiring competent DevOps professionals is a necessity when it comes to successful DevOps adoption in any organization. To achieve this, it is imperative to hire skillful DevOps consultants capable of managing the teams and building a collaborative culture.

c) Complex Infrastructure

Infrastructure complexity is yet another challenge in successful DevOps implementation as organizations find it challenging to create a common infrastructure out of different services and apps deployed in isolated environments. Educating teams on why the organization has decided to make the shift to DevOps, planning the DevOps implementation roadmap, and hiring competent DevOps consultant to go a long way in managing the complex infrastructural changes.

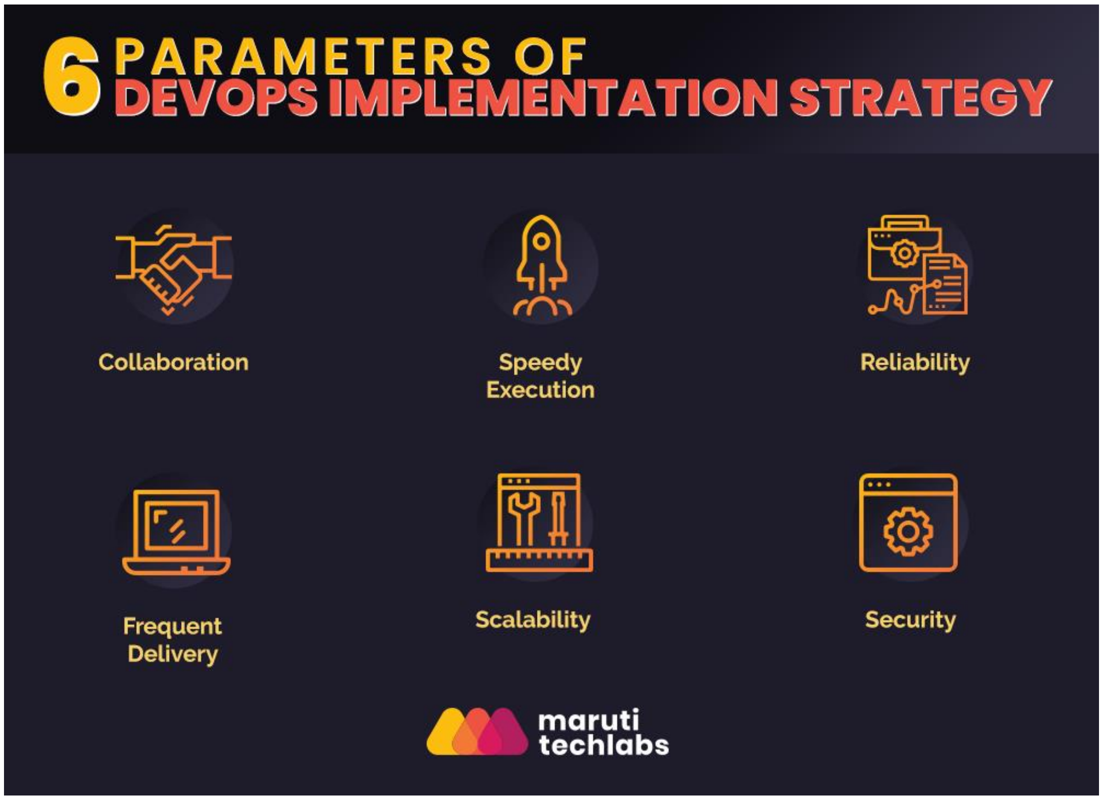
DevOps Implementation – Step-By-Step Guide

1. Build a Competent DevOps Team

The first step before you move to any new technology is the proper identification of resources and building a team competent enough to take on the challenges that come with the execution of an IT project. Some of the qualities to look for while identifying members of the DevOps team include critical thinking to find the root cause of the issue, proficiency in the latest DevOps tools and zeal to learn new ones, and an ability to troubleshoot and debug efficiently to solve the problems. A competent team can execute flawless delivery of software starting from collating requirements, planning the implementation path, and finally deploying the software.

2. Develop a Robust DevOps Strategy

The DevOps implementation strategy is essentially built on six parameters:



Parameters of a DevOps implementation

- **Speedy execution** – The ultimate objective of any organizational initiative is customer satisfaction, which is based on constant innovation and faster execution. Continuous delivery and continuous deployment of DevOps practice ensure that accuracy and speed are maintained.

- **Scalability** – Infrastructure-as-code practice assists in scalable and immaculate management of various stages (development, testing, and production) of the software product lifecycle, which is vital to DevOps success.

- **Reliability** – DevOps practices of continuous integration, continuous testing, and continuous delivery guarantee reliability of operations by ensuring safe and quality output for a positive end-user experience.

- **Collaboration** – The DevOps principle of cross-team collaboration and effective communication reduce process inefficiencies, manage time constraints, and trim the chances of project failure.

- **Frequent Delivery** – Continuous delivery, integration, and deployment practices of DevOps allow very rapid delivery cycles and minimum recovery time during implementation, leaving room for more innovation.

- **Security** – Various automated compliance policies and configuration management techniques allow the DevOps model to offer robust security through infrastructure as code and policy as code practices.

3. Start Small

It is wise to start with small initiatives before making an organizational shift to DevOps. Small-scale changes provide the benefit of manageable testing and deployment. The next steps of DevOps implementation at the organizational level should be decided based on the outcome.

4. Automate as Much as Possible

Considering the fact that fast and speedy execution lies in the backbone of DevOps, automation becomes crucial to your implementation strategy. With carefully chosen automation tools, manual hand-offs are eliminated, and processes are carried out at a faster speed, saving time, effort, and the total budget of the organization.

5. Prepare the Right Environment

For successful DevOps implementation, it is crucial to prepare the right environment of continuous testing and continuous delivery. Even a small change in the application should be tested at different phases of the delivery process. Similarly, preparing a constant delivery environment ensures that any change or addition of code is quickly deployed to production, depending on the success or failure of the automated testing.

6. Choose the Right Tools and Build a Robust Common Infrastructure

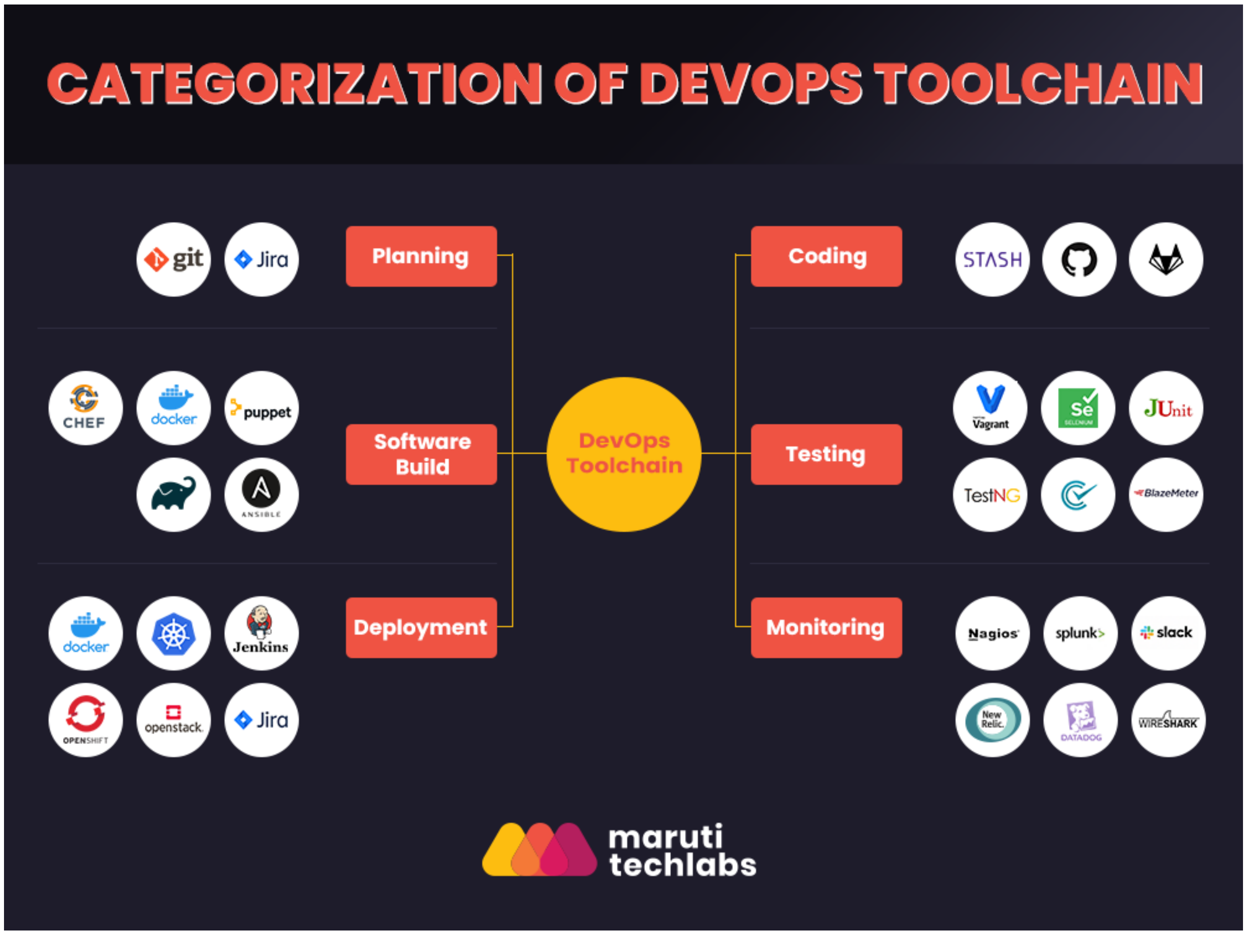
This is one of the most critical steps of the DevOps implementation process. The selection of tools should be based on their compatibility with your unique IT environment for smooth integration. The right toolset allows you to build a robust infrastructure with customized workflows and access controls, which provides enhanced usage and smooth functionality.

DevOps Toolchain

There are a number of DevOps tools that help in ensuring effective implementation; however, finding the best ones requires continuous testing and experimentation. The primary objective of these tools is to streamline and automate the different stages of the software delivery pipeline/workflow.

The DevOps toolchain can be broken down into various lifecycle stages (mentioned below) with dedicated tools for each.

- **Planning:** This is the essential phase that helps in defining business value and requirements.
  - Examples of tools — Git, Jira
- **Coding:** It involves the detailed process of software design and the creation of software code.
  - Examples of tools — Stash, GitHub, GitLab
- **Software build:** During this phase, you mainly manage various software builds and versions with the help of automated tools that assist in compiling and packaging code for future release to production.
  - Examples of tools — Docker, Puppet, Chef, Ansible, Gradle.
- **Testing:** It is the phase of continuous testing that ensures optimal code quality.
  - Example of tools — Vagrant, Selenium, JUnit, Codeception, BlazeMeter, TestNG
- **Deployment:** This is the phase of managing, scheduling, coordinating, and automating various product releases into production.
  - Examples of tools – Jenkins, Kubernetes, Docker, OpenShift, OpenStack, Jira.
- **Monitoring:** Monitoring is the phase of identifying and collecting information about different issues after software release in production.
  - Examples of tools — Nagios, Splunk, Slack, New Relic, Datadog, Wireshark



Top 12 DevOps Implementation Tools

Since no single tool works across all areas of development and delivery. The need is first to understand your processes and accordingly map the tool to establish DevOps culture in the organization successfully:

Detailed below are the top 12 DevOps tools which can be used in different phases of the software development cycle:

1. Jenkins

An excellent DevOps automation tool being adopted by an increasing number of software development teams, Jenkins is essentially an open-source CI/CD server that helps in automating the different stages of the delivery pipeline. The massive popularity of Jenkins is attributed to its enormous plugin ecosystem (more than 1000), allowing it to be integrated with a large number of other DevOps tools, including Puppet, Docker, and Chef.

Features of Jenkins

- It allows you to set up and customize CD pipeline as per individual needs.

- Runs on Windows, Linux, and macOS X, which makes it easy to get started with.
- Jenkins allows you to iterate and deploy new code with greater speed.

2. Git

Widely used across software industries, Git is a distributed SCM (source code management) DevOps tool. It allows you to easily track the progress of your development work, where you can also save different versions of source code and return to a previous one as and when required.

Features of Git

- A free and open-source tool that supports most of the version control features of check-in, merging, labels, commits, branches, etc
- Requires a hosted repository such as Github or Bitbucket that offers unlimited private repositories (for up to five team members) for free.
- Easy to learn and maintain with separate branches of source code that can be merged through Git.

3. Nagios

One of the most popular free and open-source DevOps monitoring tools, Nagiosallows you to monitor your infrastructure in real-time so that identifying security threats, detection of outages, and errors becomes easier. Nagios feeds out reports and graphs, allowing for real-time infrastructure monitoring.

Features of Nagios

- Free, open-source with various add-ons available.
- Facilitates two methods for server monitoring – agent-based and agentless.
- Allows for monitoring of Windows, UNIX, Linux, and Web applications as well.
- Available in various versions including:
    - Nagios Core – command-line tool
    - Nagios XI – web-based GUI
    - Log Server – searches log data with automatic alerts
    - Nagios Fusion – for simultaneous multiple-network monitoring

4. Splunk

Splunk is designed to make machine data usable as well as accessible to everyone by delivering operational intelligence to DevOps teams. It is an excellent choice of tool that makes companies more secure, productive, and competitive.

Features of Splunk

- Offers actionable insights with data-driven analytics on machine-generated data.

- Splunk delivers a more central and collective view of IT services.
- Quickly detects patterns, highlights anomalies, and areas of impact.

5. Docker

A forerunner in containerization, Docker is one of the most widely-used development tools of DevOps and is known to provide platform-independent integrated container security and Agile operations for cloud-native and legacy applications.

Features of Docker

- Easily automates app deployment and makes distributed development easy.
- Built-in support for Docker available by both Google Cloud and AWS.
- Docker containers support virtual machine environments and are platform-independent.

6. Kubernetes

Ideal for large teams, this DevOps tool is built on what Docker started in the field of containerization. It is a powerful tool that can group containers by logical categorization.

Features of Kubernetes

- It can be deployed through multiple computers through automated distribution.
- Kubernetes is the first orchestration container tool.
- Extremely useful in streamlining complex projects across large teams

7. Ansible

Ansible is primarily an agentless design management and organization DevOps tool. It is written in YAML. It makes it easier for DevOps teams to scale the process of automation and speed up productivity.

Features of Ansible

- Based on the master-slave architecture.
- The arrangement modules in Ansible are designated as Playbooks.
- It is an ideal DevOps tool to manage complex deployments and speed up the process of development.

8. Vagrant

Vagrant is a popular DevOps tool that can be used in conjunction with various other management tools to let developers create virtual machine environments in the same workflow. An increasing number of organizations have started using Vagrant to help transition into the DevOps culture.

Features of Vagrant

- Can work with different operating systems, including Windows, Linux, and Mac.

- Vagrant can be easily integrated and used alongside other DevOps tools such as Chef, Puppet, Ansible, etc.

9. Gradle

An extremely versatile DevOps tool, Gradle allows you to write your code in various languages, including C++, Java, and Python, among others. It is supported by popular IDEs, including Netbeans, Eclipse, and IntelliJ IDEA.

Features of Gradle

- The core model of Gradle is based on tasks – actions, inputs, and outputs.
- Gradle uses both Groovy-based DSL and a Kotlin-based DSL for describing builds.
- The incremental builds of Grade allow you to save a substantial amount of compile time.

10. Chef

Chef is a popular Ruby-based arrangement management tool that enables DevOps engineers to consider configuration management as a competitive advantage instead of a probable hurdle. The tool is mainly used for checking the configurations, and it also helps in automating the infrastructure.

Features of Chef

- Assists in standardizing and enforcing the configurations continuously.
- Chef automates the whole process and makes sure that the systems are correctly configured.
- Chef helps you ensure that the configuration policies remain entirely flexible, readable, and testable.

11. Worksoft

Worksoft is another popular DevOps tool that offers incredible support for both web and cloud applications. It has a robust ecosystem of solutions for various enterprise applications spanning across the entire pipeline of continuous delivery.

Features of Worksoft

- Capable of integrating UI and end-to-end testing into the CI pipeline, thus speeding the process.
- Allows medium and large-scale businesses to create risk-based continuous testing pipelines that feed into application production environments for scalability.
- Offers integrations with various third-party solutions to allow the companies to choose tools best suited for their individual, organizational needs and seamlessly manage tasks across the entire DevOps release cycle.

12. Puppet

Puppet is an open-source configuration management tool that is used for deploying, configuring and managing servers.

Features of Puppet

- It offers master-slave architecture.
- Puppet works smoothly for hybrid infrastructure and applications.
- Compatible with Windows, Linux, and UNIX operating systems.

The DevOps approach is here to stay, and it will continue to be implemented by enterprises increasingly in the future. Recent research conducted by Technavioestimated a whopping 19% CAGR (Compound Annual Growth Rate) in the global DevOps market (from 2016–2020), highlighting the goldmine of benefits implementing DevOps holds.

To ensure the successful implementation of the DevOps process, it is essential to plan out a solid DevOps strategy and select DevOps tools that fit in well with other tools and the development environment. Finding the best DevOps tools takes some amount of testing and a level of experimentation. It mainly takes more time to set up and configure open-source tools. The world of DevOps is rife with an outstanding and unique open-source tool - we found the tools listed above to be the best in class and think they should be included in every DevOps engineer's shortlist. What are open-source DevOps tools in your toolkit?

The article initially published on Maruti Techlabs blog.