**Deliver apps faster and save money with Red Hat OpenShift**

## An ROI of 531% over 5 years? It’s possible with Red Hat OpenShift.

Organizations are increasingly relying on software and new applications to be the foundation of new initiatives. To keep up with these expectations, and customer demands, traditional technology approaches aren’t able to deliver innovation fast enough.

Meanwhile, entire industries are seeing greater competition that requires IT to be more agile to succeed. New technologies, new architectures, automation, and analytics all rely on the foundation of a modern development platform.

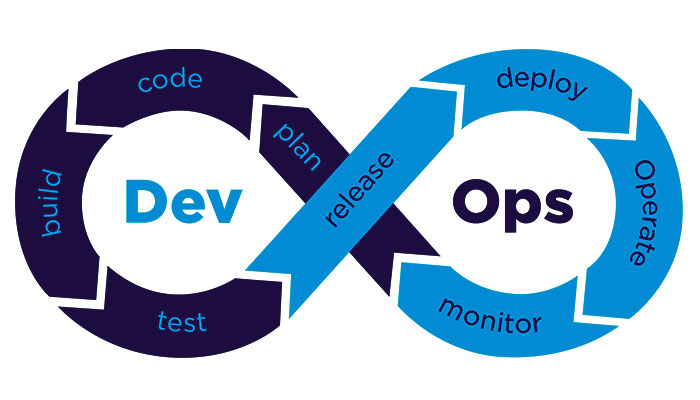
IDC interviewed 9 organizations using Red Hat® OpenShift as their primary app development platform. Findings from this research showed that those organizations saw significant value through OpenShift in delivering timely, compelling applications and features across their environments. And the platform supports key It initiatives, like containers, microservices, and cloud migration strategies.

**Organizations using Red Hat OpenShift, interviewed by IDC, have seen impressive benefits, including:**

* 531% average 5-year return on investment
* 66% faster application development life cycles
* 35% less IT staff time required per app developed
* 38% lower IT infrastructure and development platform costs per app

# Introduction to DevOps

There is already a lot of buzz doing rounds about what exactly is DevOps? Though there is no common consensus on its definition, but the philosophy and practices of DevOps explain a lot about it.



The term “DevOps” was coined in the year 2009 by Partick Debois. As its name implies it is a conjunction of two different words, “Development” and “Operations”. The reason behind this name is that it combines these two areas of an organization. Unlike the conventional organizations where these two areas were considered separate, DevOps believes in bridging this gap by incorporating their operations with one another at each step. By means of combining, DevOps helps achieve agility, quality and consistency at the same time.

Fundamentally the concept of DevOps revolves around demolishing the barriers between the traditionally siloed teams by changing the culture and mindset. Conventionally teams were highly demarcated with respect to the job responsibilities, which resulted in rigidity and slow work process. Communication and resource inefficiency acted as catalysts to the situation. However DevOps believes in optimizing the output by coordinating development and operations.

Now the job roles are not confined to the department, but the professionals go beyond their supposed roles to satisfy the customers’ needs. And this is possible only when the entire staff is dedicated to perform the single organizational goal.

Quickly releasing a new product or new features to the market is a challenging task for the organizations worldwide. The toughest job is to bring together all the distinct groups such as Development, Quality Assurance and IT Operations to get the work done in a quickest way to release the product as soon as possible.

It has been seen that processes and practices are kept on evolving to overcoming those challenges. Just a decade ago there is no such term called DevOps exists in our corporate world, but later in 2009, a methodology defined a set of processes to **collaborate and communicate the Development, Quality assurance, and IT operations to reduce the time to market** the product significantly, which is popularly called as DevOps.

The DevOps methodology is a set of practices designed to overcome the gap between development, QA and Operations by effective communication and collaboration, incorporating **continuous integration process with automated deployment.** The purpose is to set up an environment to develop, test and release the software product to the market quickly and reliably.

### Traditional Approach Vs. DevOps Approach

In a traditional waterfall method, the developer writes the codes as per the requirements in their local environment. Once the product is built, the quality assurance (QA) tests the product in an environment similar to the production environment. Finally, when the desired requirements are met, the product is released to the operations for use. The whole process takes lots of time from taking the requirements till the deployment of the product to the operations.  As both the groups, development and operations works independently, the chances are high when the product is finally reached to the operations after a longer duration. Also, the products may not work as expected or may face some roadblock.



While, the DevOps process has an improved approach to deal with those situations. It emphasizes on communication and collaboration of the development team, QA team and the operations team to perform **continuous development, continuous integration, continuous delivery and monitoring the processes continuously** by taking the help of various tools, which fills the big gap in between them and expedite the work for a faster release. The faster release to the operations enables the organizations to quickly respond to the changes in the business requirements.

### Relationship Between Agile and DevOps

DevOps comes into existence considering one of the important benefits of Agile Software development method which enables the organization to release the product quickly. But further additions of more processes differentiate DevOps from Agile.

Agile principles are only applied to the Development and QA processes, which believes in creating small teams to build and release working software within a short duration, called as sprint.  The team only focuses on the sprint and there is no communication with the operations.

While the DevOps more focused on uninterrupted engagement of development, QA and Operations teams throughout the development life cycle. The Operation team should continuously participate with development team while discussing on the project goals, high-level released timelines and other business needs. From the beginning, the operation team should provide the operation related requirements to the development team, and also need to validate them. A continuous monitoring of the project with constant and effective communications allows quick turn around.

### What you will get from DevOps?

Let us discuss some of the benefits that we can get from DevOps cultures and processes.

* The foremost benefit we will get that, it **reduces the time to market** for the product significantly by bridging the gap between responsible teams and following the continuous development process
* The better synchronization between the teams provides the team members a **transparent vision** on the in-progress work, thus they can see the issues or roadblocks before that actually comes. That’s help them to plan in a better way to overcome such issues.
* Due to the transparency in this process, there is a **sense ownership** for the developers while developing the product. They actually own their code from conception to operation.
* The **automated deployment** of the product using automation tools on multiple environments allows you to **quickly identify the environment related issues**. Normally, when the environments are different, it takes a lot of time to single out the environment related issues.

### Multiple Tools For DevOps

As DevOps is the collaboration of Development, QA and Operations, it is obvious that a **single tool cannot be adequate for all the needs.** So there are **multiple tools required** in each stage to perform all the operations successfully.

Let’s have a look on few of those tools can be used in different stages.

* Tools for Monitoring – Nagios, NewRelic, Graphite etc.
* Virtualization and Containerization Tools  – Vagrant, VMware, Xen, Docker etc.
* Tools for build, test and deployment – Jenkins, Maven, Ant, Travis, Bamboo, Teamcity etc.
* For configuration Management – Puppet, Chef, Ubuntu Juju, Ansible, cfengine etc.
* For Orchestration – Zookeeper, Noah etc.
* For Cloud services – Azure, Openstack, Rackspace etc.

There are other such tools which can be used for code merging, version control etc. to effectively apply the DevOps processes.

### Myths Surroundings DevOps

Though DevOps comes into existence decades ago, but still it is a new term for many of us. So, it is obvious that misconceptions can happen.

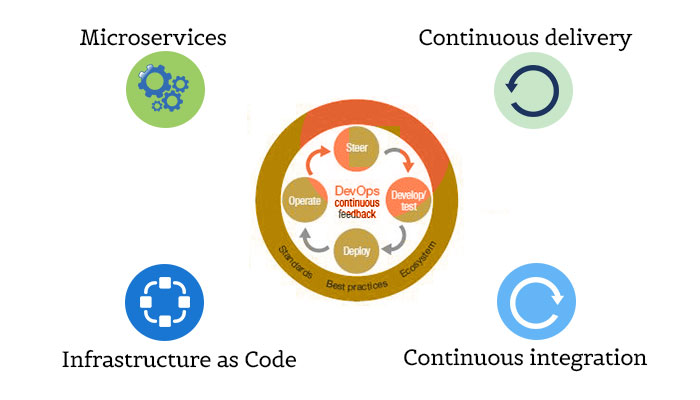
Some people think that implementing DevOps allows the developers to takes away operations job. This is completely not true. DevOps emphasizes on collaborative efforts from both the team, hence the **developers having skills on operations as well will get the higher advantage in the fact paced business cycle.**

It is also widely believed that DevOps can be practiced by just implementing a set of tools for all the works. That’s not true; **just by using some tools we cannot achieve DevOps**, but by practicing its process and wisely using the tools can help us to achieve its core values.

The DevOps is becoming popular day by day due to its culture of continuous development, continuous integration, continuous delivery and continuous monitoring processes by collaborating the Development and Operations to reduce the time to market significantly.

# DevOps Best Practices

DevOps strive to fasten the processing speed of operations as well as optimize the overall outcome. For this DevOps promote certain key practices –



* **Continuous integration –** This primary practice emphasizes on merging the changes in software codes and storing it into the repository. This activity is followed by the execution of tests and automated builds. By doing this DevOps seeks to find the potential errors quickly and improve the software quality at the end.
* **Continuous delivery –** DevOps stresses to build, test and release the software product automatically in lesser time. This requires the development and testing environments to be tightly integrated. With this practice the developers always have the code constructs at their disposal which is already tested.
* **Microservices –** This is a software development practice where a big application is made up of small services dedicated to perform a single operation and connected through a light-weight HTTP API. These microservices can be developed in any programming language and can be deployed independently.
* **Infrastructure as Code –** Here the infrastructure is maintained with the help of programming. If the infrastructure is built on the cloud, it can be accessed as the

### Terminology

IaaS: Infrastucture as a Service

PaaS: Platform as a Service

Saas:

### Processes

Build Automation

Continuous Integration and Continuous Deployment

### Tools

Jenkins

Chef and Puppet

Docker