DEVOPS

1. What is DevOps: DevOps Core, Working, and Uses explained

DevOps represents an era of a dynamic shift in the IT culture, which has brought the developmental and operational team together. Generally, it is the combination of philosophies and tools that improves the team's capabilities of producing results at a higher efficacy.

The core of DevOps revolves around Agile practices. Moreover, it has even engaged the best ESM or Enterprise Systems management. The typical process engaged in the DevOps approach is automation and Agile methodologies. It is for software developers, operators, and several segments of system administrators who aim to procure efficient and high-quality results. For in-depth understanding, read our 'DevOps tutorial.'

### 2. Best DevOps Tools to Learn and Master

Bridging the gap between the traditional development and operational teams, DevOps use an [Agile methodology](https://www.simplilearn.com/tutorials/agile-scrum-tutorial/what-is-agile) that amalgamates the two teams and prompts them to work in collaboration.

The efficient DevOps tools like Ansible, [Docker](https://www.simplilearn.com/tutorials/docker-tutorial/what-is-docker), Puppet, [Chef](https://www.simplilearn.com/tutorials/chef-tutorial/what-is-chef), [Git](https://www.simplilearn.com/tutorials/git-tutorial/what-is-git), Spunk, Nagios, and [Jenkins](https://www.simplilearn.com/tutorials/jenkins-tutorial/what-is-jenkins) alike make the automation task easygoing.

The two biggest parts of DevOps are automation and continuity. It is the organizational needs that determine which tool is best for the particular software development. In this DevOps tutorial, you will know about the advanced tools and technology that seamlessly make organizational processes.

### 3. Kubernetes vs. Docker: Know the Major Differences

Being in touch with new and innovative technology is quite challenging for companies. The DevOps engineer's major role is to remain in touch with such tools that surround new technology.

The two major tools of DevOps are Kubernetes and Docker. Both are open-source container tools and have basic differences in their setting and operation.

The list of differences is classified into different headings like the concept, networking, GUI, installation, load balancing, container setup, scalability and autoscaling, logging and monitoring, rolling updates and rollbacks, and many more.

One question that pops our mind is, which is better? Well, it depends upon the organizational requirements.

### 4. How to Become a DevOps Engineer: Roles, Responsibilities, and Skills Required

In any company, the development team creates a product while the operational team maintains and manages it. DevOps is the platform that ensures collaboration of the development and operation team of the organization.

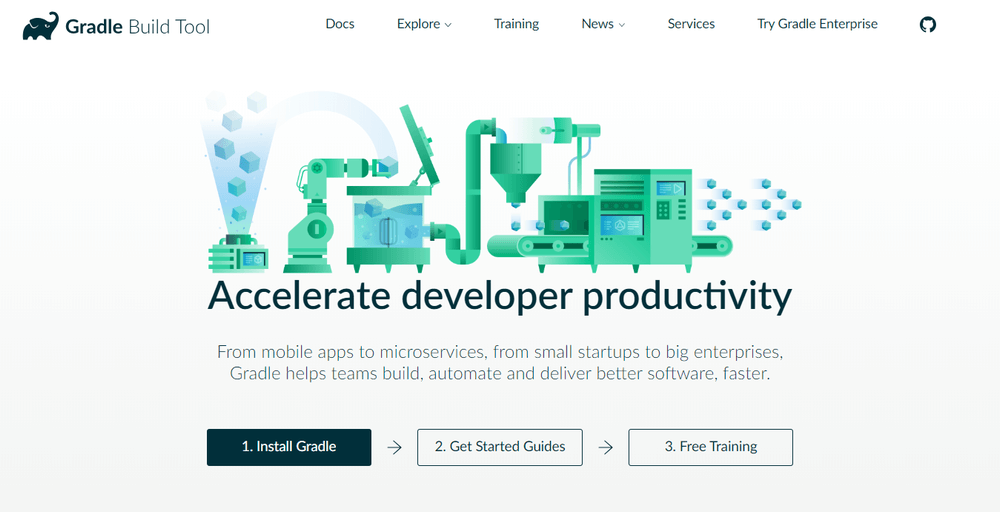
[DevOps Engineer](https://www.simplilearn.com/tutorials/devops-tutorial/how-to-become-devops-engineer) is an IT professional who has a deep understanding of the entire lifecycle of software development. They research and analyze why the organization needs a shift in the model and the need for software creation. The salary of the engineers depends upon their certification and experience.

# Why DevOps Is Important (5 Reasons)

**Published On: June 28, 2017 by**[**Shadow-Soft Team**](https://shadow-soft.com/author/shadow-soft/)

DevOps describes a culture and set of processes that bring development and operations teams together to complete software development. It allows organizations to create and improve products at a faster pace than they can with traditional software development approaches. And, it’s gaining popularity at a rapid rate.

## 1. Gradle



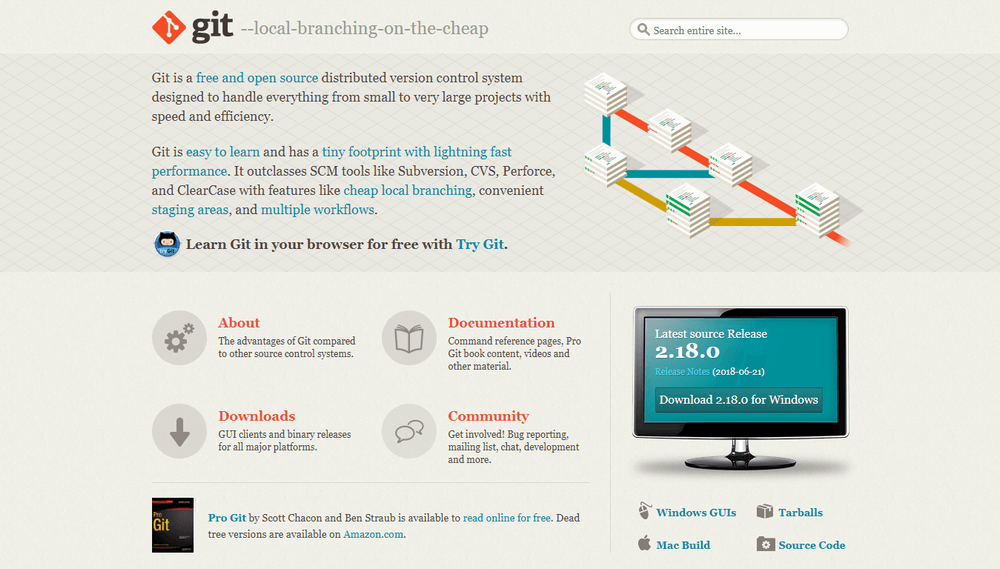
Your DevOps tool stack will need a reliable build tool. Apache Ant and Maven dominated the automated build tools market for many years, but [Gradle](https://gradle.org/) showed up on the scene in 2009, and its popularity has steadily grown since then. Gradle is an incredibly versatile tool which allows you to write your code in Java, C++, Python, or other languages. Gradle is also supported by popular IDEs such as Netbeans, Eclipse, and IntelliJ IDEA. If that doesn’t convince you, it might help to know that Google also chose it as the [official build tool](https://developer.android.com/studio/build/) for Android Studio.

While Maven and Ant use XML for configuration, Gradle introduces a Groovy-based DSL for describing builds. In 2016, the Gradle team also released a [Kotlin-based DSL](https://github.com/gradle/kotlin-dsl), so now you can write your build scripts in Kotlin as well. This means that Gradle does have some learning curves, so it can help a lot if you have used Groovy, Kotlin or another JVM language before. Besides, Gradle uses Maven’s repository format, so dependency management will be familiar if you have prior experience with Maven. You can also [import your Ant builds](https://docs.gradle.org/current/userguide/ant.html) into Gradle.

The best thing about Gradle is [incremental builds](https://blog.gradle.org/introducing-incremental-build-support), as they save a nice amount of compile time. According to Gradle’s [performance measurements](https://gradle.org/gradle-vs-maven-performance/), it’s up to 100 times faster than Maven. This is in part because of incrementality, but also due to Gradle’s [build cache](https://blog.gradle.org/introducing-gradle-build-cache) and [daemon](https://docs.gradle.org/current/userguide/gradle_daemon.html). The build cache reuses task outputs, while the Gradle Daemon keeps build information hot in memory in-between builds.

All in all, Gradle allows faster shipping and comes with a lot of configuration possibilities.

## 2. Git



[Git](https://git-scm.com/) is one of the most popular DevOps tools, widely used across the software industry. It’s a distributed SCM (source code management) tool, loved by remote teams and open source contributors. Git allows you to track the progress of your development work. You can save different versions of your source code and return to a previous version when necessary. It’s also great for experimenting, as you can create separate branches and merge new features only when they’re ready to go.

To integrate Git with your DevOps workflow, you also need to host repositories where your team members can push their work. Currently, the two best online Git repo hosting services are [GitHub](https://github.com/) and [Bitbucket](https://bitbucket.org/). GitHub is more well-known, but Bitbucket comes with free unlimited private repos for small teams (up to five team members). With GitHub, you get access only to public repos for free—which is still a great solution for many projects.

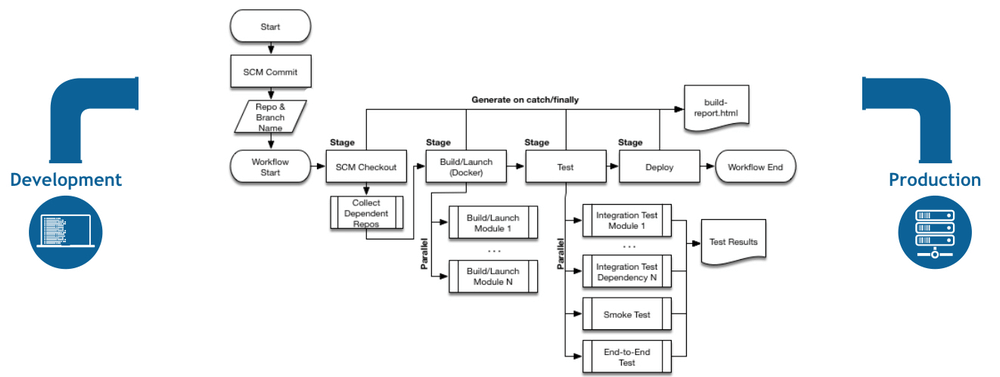
Both GitHub and Bitbucket have fantastic integrations. For example, you can integrate them with Slack, so everyone on your team gets notified whenever someone makes a new commit.

## 3. Jenkins



[Jenkins](https://jenkins.io/) is the go-to DevOps automation tool for many software development teams. It’s an open source CI/CD server that allows you to automate the different stages of your delivery pipeline. The main reason for Jenkins’ popularity is its huge plugin ecosystem. Currently, it offers [more than 1,000 plugins](https://plugins.jenkins.io/), so it integrates with almost all DevOps tools, from Docker to Puppet.

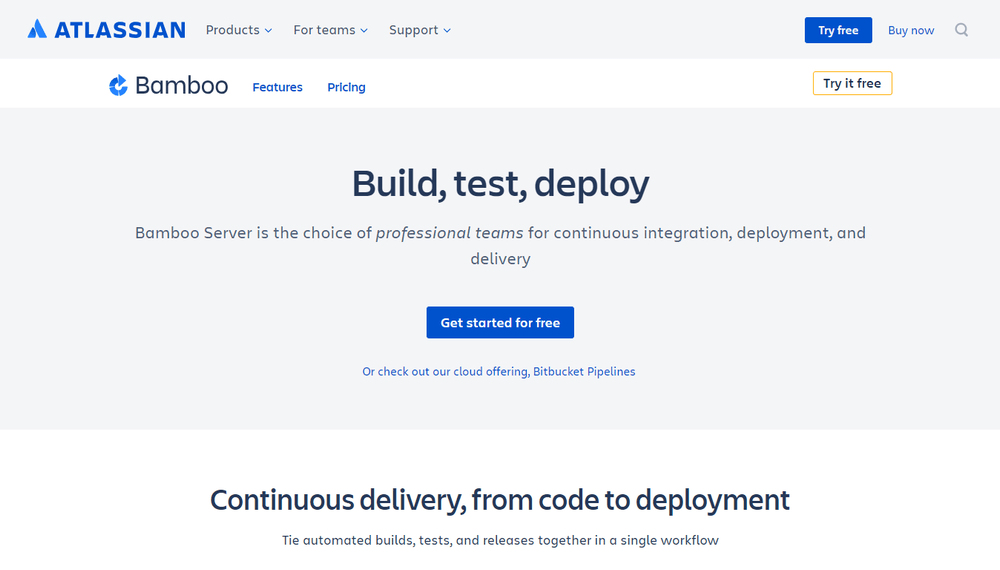
With Jenkins, you can set up and customize your CI/CD pipeline according to your own needs. I found the following example in the [Jenkins Docs](https://jenkins.io/doc/book/pipeline/). And, this is just one of the possibilities. Nice, isn’t it?



It’s easy to [get started with Jenkins](https://jenkins.io/download/), as it runs out-of-the-box on Windows, Mac OS X, and Linux. You can also easily [install it with Docker](https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+with+Docker). You can set up and configure your Jenkins server through a web interface. If you are a first-time user, you can choose to install it with frequently used plugins. However, you can create your own custom config as well.

With Jenkins, you can iterate and deploy new code as quickly as possible. It also allows you to measure the success of each step of your pipeline. I’ve heard people complaining about Jenkins’ “ugly” and non-intuitive UI. However, I could still find everything I wanted without any problem.

## 4. Bamboo

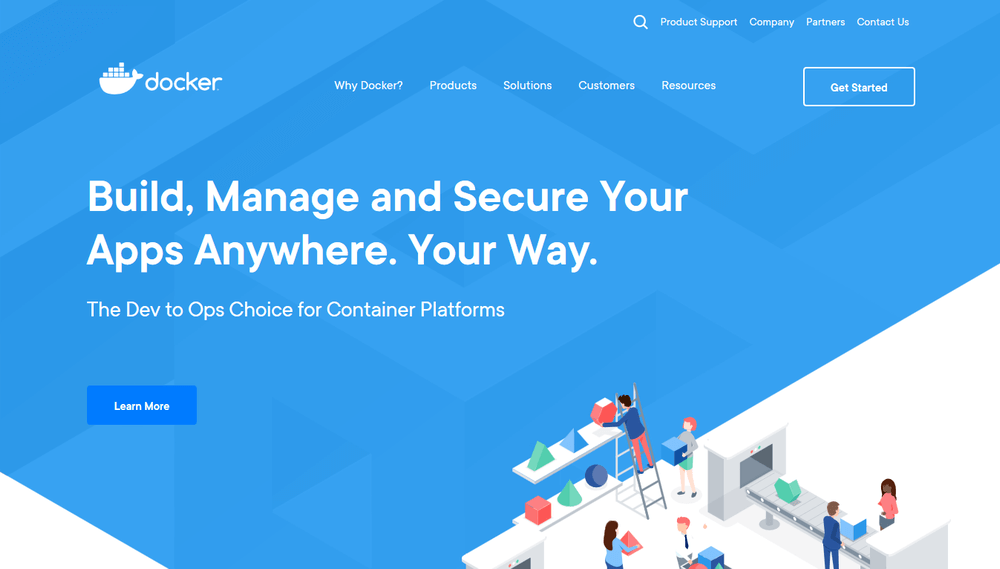


[Bamboo](https://www.atlassian.com/software/bamboo) is Atlassian’s CI/CD server solution that has many similar features to Jenkins. Both are popular DevOps tools that allow you to automate your delivery pipeline, from builds to deployment. However, while Jenkins is open source, Bamboo comes with a price tag. So, here’s the eternal question: is it worth choosing proprietary software if there’s a free alternative? It depends on your budget and goals.

Bamboo has many pre-built functionalities that you have to set up manually in Jenkins. This is also the reason why Bamboo has fewer plugins (around 100 compared to Jenkins’ 1000+). In fact, you don’t need that many plugins with Bamboo, as it does many things out-of-the-box.

Bamboo seamlessly integrates with other Atlassian products such as Jira and Bitbucket. You also have access to built-in Git and Mercurial branching workflows and test environments. All in all, Bamboo can save you a lot of configuration time. It also comes with a more intuitive UI with tooltips, auto-completion, and other handy features.

## 5. Docker

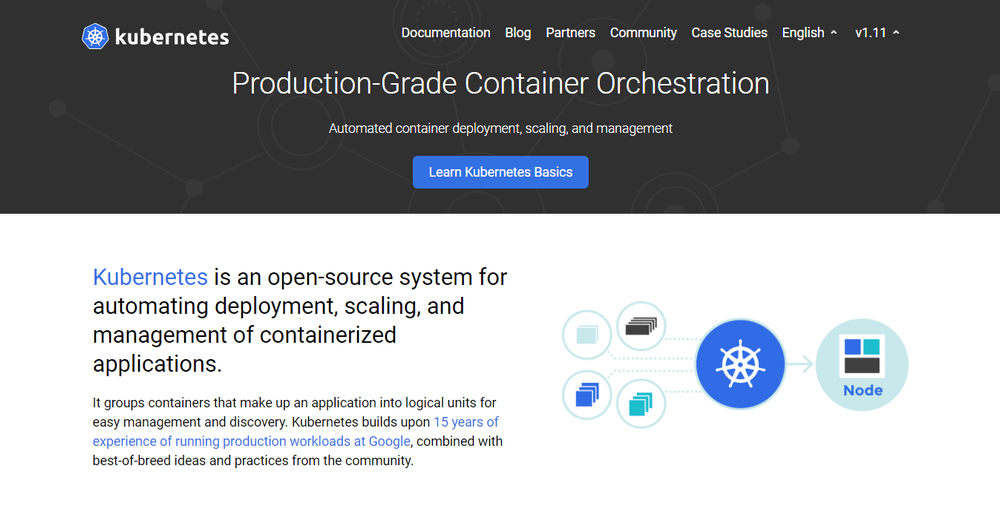


[Docker](https://www.docker.com/) has been the number one container platform since its launch in 2013 and continues to improve. It’s also thought of as one of the most important DevOps tools out there. Docker has made containerization popular in the tech world, mainly because it makes distributed development possible and automates the deployment of your apps. It isolates applications into separate containers, so they become portable and more secure. Docker apps are also OS and platform independent. You can use Docker containers instead of virtual machines such as VirtualBox.

What I like the most about Docker is that you don’t have to worry about dependency management. You can package all dependencies within the app’s container and ship the whole thing as an independent unit. Then, you can run the app on any machine or platform without a headache.

Docker integrates with [Jenkins](https://jenkins.io/solutions/docker/) and [Bamboo](https://confluence.atlassian.com/bamboo/getting-started-with-docker-and-bamboo-687213473.html), too. If you use it together with one of these automation servers, you can further improve your delivery workflow. Besides, Docker is also great for cloud computing. In recent years, all major cloud providers such as AWS and Google Cloud added support for Docker. So, if you are planning a cloud migration, Docker can ease the process for you.

## 6. Kubernetes



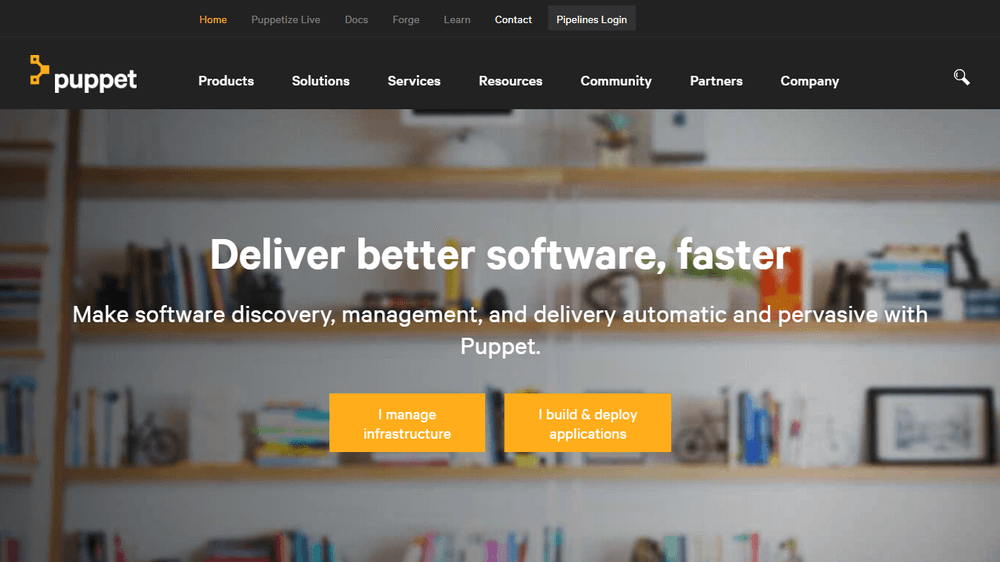
This year, everyone is talking about [Kubernetes](https://kubernetes.io/). It’s a container orchestration platform that takes containerization to the next level. It works well with Docker or any of its alternatives. Kubernetes is still very new; its first release came out in 2015. It was founded by a couple of Google engineers who wanted to find a solution to manage containers at scale. With Kubernetes, you can group your containers into logical units.

You may not need a container orchestration platform if you have just a few containers. However, it’s the next logical step when you reach a certain level of complexity and need to scale your resources. Kubernetes allows you to automate the process of managing hundreds of containers.

With Kubernetes, you don’t have to tie your containerized apps to a single machine. Instead, you can deploy it to a cluster of computers. Kubernetes automates the distribution and scheduling of containers across the whole cluster.

A Kubernetes cluster consists of one master and several worker nodes. The master node implements your pre-defined rules and deploys the containers to the worker nodes. Kubernetes pays attention to everything. For instance, it notices when a worker node is down and redistributes the containers whenever it’s necessary.

## 7. Puppet Enterprise

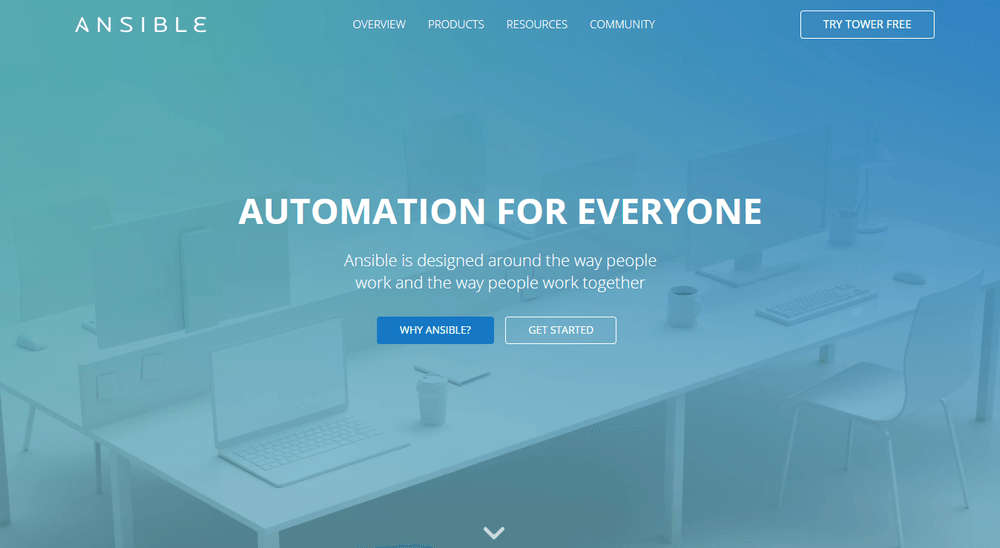


[Puppet Enterprise](https://puppet.com/products/puppet-enterprise) is a cross-platform configuration management platform. It allows you to manage your infrastructure as code. As it automates infrastructure management, you can deliver software faster and more securely. Puppet also provides developers with an [open-source tool](https://puppet.com/download-open-source-puppet) for smaller projects. However, if you are dealing with a larger infrastructure, you may find value in Puppet Enterprise’s [extra features](https://puppet.com/products/why-puppet/puppet-enterprise-and-open-source-puppet), such as:

* Real-time reports
* Role-based access control
* Node management

With Puppet Enterprise, you can manage multiple teams and thousands of resources. It automatically understands relationships within your infrastructure. It deals with dependencies and handles failures smartly. When it encounters a failed configuration, it skips all the dependent configurations as well. The best thing about Puppet is that it has [more than 5,000 modules](https://forge.puppet.com/) and integrates with [many popular DevOps tools](https://puppet.com/products/managed-technology).

## 8. Ansible

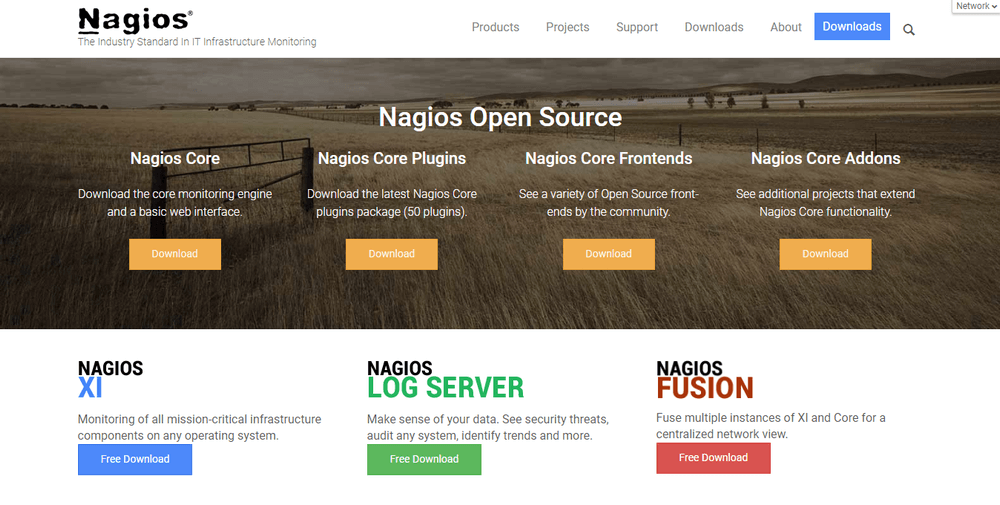


[Ansible](https://www.ansible.com/) is a configuration management tool, similar to Puppet and Chef. You can use it to configure your infrastructure and automate deployment. Its main selling points compared to other similar DevOps tools are simplicity and ease of use. Ansible follows the same Infrastructure As Code (IAC) approach as Puppet. However, it uses the super simple YAML syntax. With Ansible, you can define tasks in YAML, while Puppet has its own declarative language.

Agentless architecture is another frequently mentioned feature of Ansible. As no daemons or agents run in the background, Ansible is a secure and lightweight solution for configuration management automation. Similar to Puppet, Ansible also has [several modules](https://docs.ansible.com/ansible/latest/modules/modules_by_category.html).

If you want to better understand how Ansible fits into the DevOps workflow take a look at [this post](https://www.redhat.com/en/blog/integrating-ansible-jenkins-cicd-process) by the Red Hat Blog. It shows how to use Ansible for environment provisioning and application deployment within a Jenkins pipeline.

## 9. Nagios



[Nagios](https://www.nagios.org/) is one of the most popular free and open source DevOps monitoring tools. It allows you to monitor your infrastructure so that you can find and fix problems. With Nagios, you can keep records of events, outages, and failures. You can also keep an eye on trends with the help of Nagios’ graphs and reports. This way, you can forecast outages and errors and detect security threats.

Although there are many DevOps tools for infrastructure monitoring, Nagios stands out due to its rich [plugin ecosystem](https://exchange.nagios.org/). As Nagios has been around for a while (since 2002), there’s a vast community around it. Besides plugins, they also make add-ons, tutorials, translations, and other goodies—all for free.

Nagios offers four open source monitoring solutions:

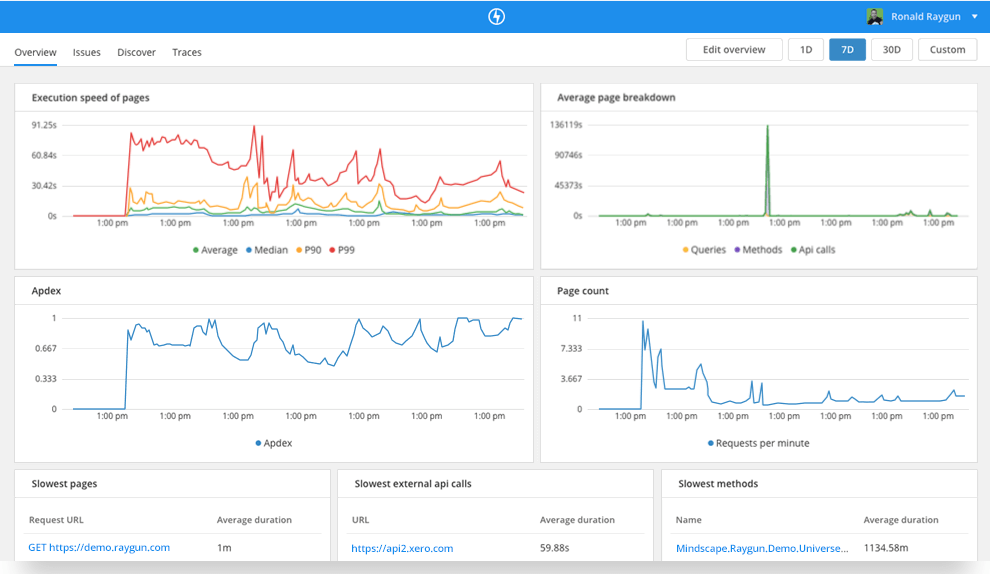
1. Nagios Core
2. Nagios XI
3. Nagios Log Server
4. Nagios Fusion

**Nagios Core** is a command line tool, with all the basic functionalities. You can also opt for **Nagios XI** that comes with a web-based GUI and monitoring wizard. Nagios writes a handy [comparison of their capabilities](https://www.nagios.org/downloads/nagios-core/).

**Nagios Log Server** lets you search log data and set up alerts about potential threats. And, **Nagios Fusion** allows you to monitor multiple networks at the same time.

On the whole, Nagios provides DevOps teams with an infrastructure monitoring solution. However, it can take a while to set it up and make it compatible with your environment.

## 10. Raygun

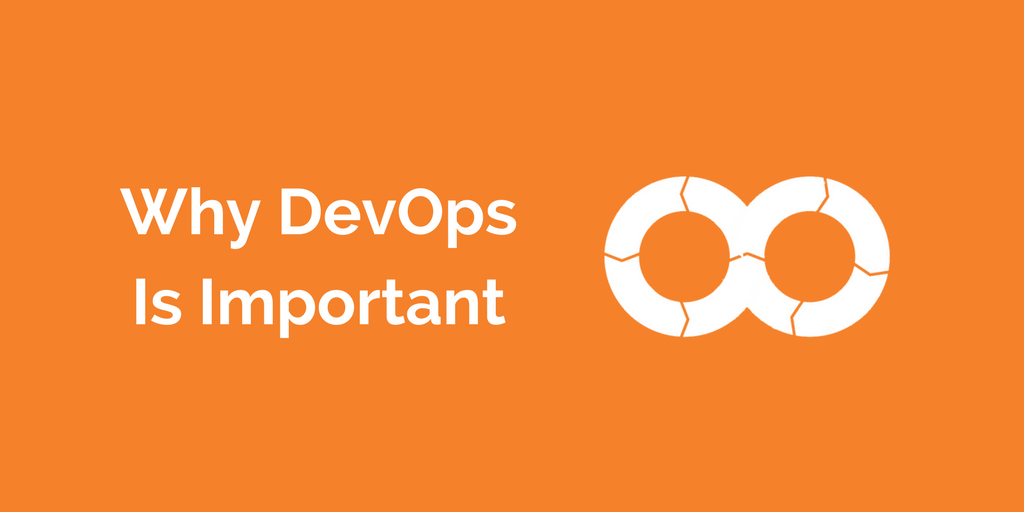


[Raygun](https://raygun.com/) is a world-class error monitoring and crash reporting platform. [Application performance monitoring](https://raygun.com/platform/apm) (APM) is its most recent product. Raygun’s DevOps tool helps you diagnose performance issues and tracking them back to the exact line of code, function, or API call. The APM tool also fits well with Raygun’s error management workflow. For example, it automatically identifies your highest priority problems and creates issues for you.

Raygun APM can help you make the most out of other DevOps tools, as you are always notified about the problems. Since it automatically links errors back to the source code, Raygun brings Development and Operations together by providing one source of truth for the whole team the cause of errors and performance problems.

## Which DevOps tools are right for your team?

Finding the best DevOps tools takes some testing and experimentation. It usually takes more time to set up and configure open-source tools. Most commercial DevOps tools come with free trials that allow you to test and evaluate them at no cost. It all boils down to your needs and goals.



According to [DevOps.com](https://devops.com/state-devops-adoption-trends-2017/), the adoption rate increased significantly from 2015 to 2016.

* **2015:** 66 percent of organizations were adopting DevOps, 19 percent weren’t adopting DevOps, and 15 percent were undecided.
* **2016:** 74 percent of organizations were adopting DevOps, 16 percent weren’t adopting DevOps, and 10 percent were undecided.

Here are the top five reasons why the industry has been so quick to adopt DevOps principles:

## **1. Shorter Development Cycles, Faster Innovation**

When development and operations teams are in separate silos, it’s usually difficult to tell if an application is ready for operations. When development teams simply turn over an application, the operations’ cycle times are extended needlessly.

With a combined development and operations team, applications are ready for use much more quickly. This is important, since companies succeed based on their ability to innovate faster than their competitors do. In fact, Kevin Murphy from [Red Hat](https://www.redhat.com/) estimates that shorter development cycles translate to bringing an application to market 60 percent faster than with traditional approaches.

## **2. Reduced Deployment Failures, Rollbacks, and Time to Recover**

Part of the reason teams experience deployment failures is due to programming defects. The shorter development cycles with DevOps promote more frequent code releases. This, in turn, makes it easier to spot code defects. Therefore, teams can reduce the number of deployment failures using agile programming principles that call for collaboration and modular programming. Rollbacks are similarly easier to manage because, when necessary, only some modules are affected.

Time to recover is an important issue, because some failure has to be expected. But recovery is much faster when the development and operations teams have been working together, exchanging ideas and accounting for both teams’ challenges during development.

## **3. Improved Communication and Collaboration**

DevOps improves the software development culture. Combined teams are happier and more productive. The culture becomes focused on performance rather than individual goals. When the teams trust each other, they can experiment and innovate more effectively. The teams can focus on getting the product to market or into production, and their KPIs should be structured accordingly.

It’s no longer a matter of “turning over” the application to operations and waiting to see what happens. Operations doesn’t need to wait for a different team to troubleshoot and fix a problem. The process becomes increasingly seamless as all individuals work toward a common goal.

## **4. Increased Efficiencies**

Increased efficiency helps to speed the development process and make it less prone to error. There are ways to automate DevOps tasks. Continuous integration servers automate the process of testing code, reducing the amount of manual work required. This means that software engineers can focus on completing tasks that can’t be automated.

Acceleration tools are another opportunity for increasing efficiency. For example:

* Scalable infrastructures, such as cloud-based platforms, increase the access the team has to hardware resources. As a result, testing and deployment operations speed up.
* Build acceleration tools can be used to compile code more quickly.
* Parallel workflows can be embedded into the continuous delivery chain to avoid delays; one team waits for another to complete its work.
* Using one environment avoids the useless task of transferring data between environments. This means you don’t have to use one environment for development, a different environment for testing, and a third for deployment.

## **5. Reduced Costs and IT Headcount**

All of the DevOps benefits translate to reduced overall costs and IT headcount requirements. According to Kevin Murphy from Red Hat, DevOps development teams require 35 percent less IT staff and 30 percent lower IT costs.