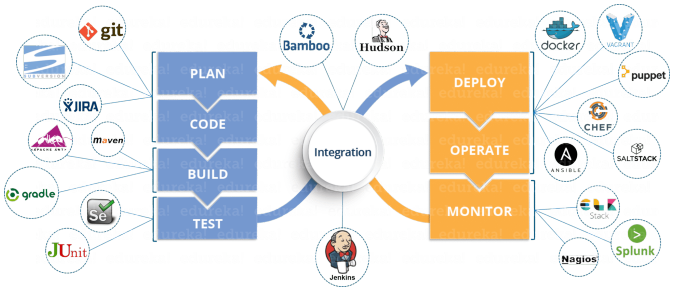
DevOps Tools

Table of Contents

DevOps

Before going any further, let's recap what are the different tools and where they fall in the DevOps lifecycle.



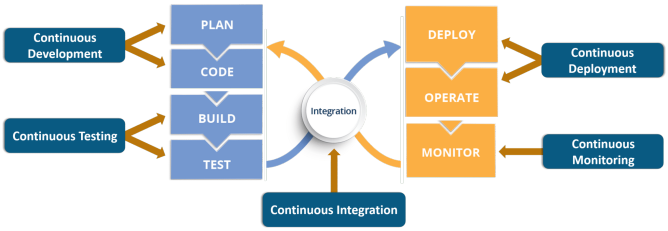
Most Used DevOps Tools

DevOps Lifecycle Phases (DevOps Tools)

Well i'm pretty sure you're impressed with the above image. But, you might still have problems relating the tools to various phases. Don't you?

In that case, let's take a step back and first understand what are the various phases present in the DevOps lifecycle. Below are the 5 different phases any software/ application has to pass through, when developed via the DevOps lifecycle:-

1. Continuous Development
2. Continuous Testing
3. Continuous Integration
4. Continuous Deployment
5. Continuous Monitoring



Phases explained - DevOps tools

1. Continuous Development

This is the phase which involves 'planning' and 'coding' of the software application's functionality.

There are no tools for planning as such, but there are a number of tools for maintaining the code.

The vision of the project is decided during the ‘planing’ phase and the when they start writing the code, the act is referred to as ‘coding’ phase.

The code can be written in any language, but it is maintained by using Version Control tools. These are the Continuous Development DevOps tools. The most popular tools used are:

1. Git,
2. SVN,
3. Mercurial,
4. CVS and JIRA.

So why is it important to main versions of the code? Which of the Dev vs Ops problem does it solve? Let’s understand that first.

Versions are maintained (in a central repository), to hold a single source of truth. So that all the developers can collaborate on the ‘latest committed’ code, and even operations can have access to that same code when they plan to make a release. Whenever a mishap happens during a release, or even if there are lots of bugs in the code (faulty feature), there is nothing to worry. Ops can quickly rollback the deployed code and thus revert back to the previous stable state. So, which is my favorite tool? That has got to be Git & GitHub. Why? Because Git allows developers to collaborate with each other on a Distributed VCS (Version Control System).

Since there is no dependency on the central server, ‘pulls’ & ‘pushes’ to the repository can be made from remote locations. This central repository where the code is maintained is called GitHub.

git - devops tools image::./img/devops-git.png[Devops,1024,480,pdfwidth=50%,scaledwidth=50%,float="right",align="center"]

Git is in-fact the world’s leading Version Control system. If you don’t want to take my word for it, you can just google that up. So let’s move on to the next topic in this DevOps tools blog. You can read more about Git from here: What is Git?

2. Continuous Testing

When the code is developed, it is maddening to release it straight to deployment. The code should first be tested for bugs and performance. Can we agree on that statement?

If yes, then what would be the procedure to perform the tests? Would it be manual testing? Well, it can be, but it is very inefficient. So, what is better? Automation testing? Exactly! Sounds amazing right?

Automation testing is the answer to a lot of cries of manual testers. Tools like Selenium, TestNG, JUnit/ NUnit are used to automate the execution of our test cases. So, what are its benefits?

Automation testing saves a lot of time, effort and labor for executing the tests manually. Besides that, report generation is a big plus. The task of evaluating which test cases failed in a test suite gets

simpler. These tests can also be scheduled for execution at predefined times. Brilliant right? And the continuous use of these tools while developing the application is what forms the 'Continuous Testing' phase during DevOps lifecycle. Which of these is my favorite tool? A combination of these tools actually!

Selenium is my favorite, but Selenium without TestNG is equivalent to a snake without a poisonous sting, atleast from the perspective of DevOps lifecycle.

Selenium does the automation testing, and the reports are generated by TestNG. But to automate this entire testing phase, we need a trigger right? So, what is the trigger? This is where the role of Continuous Integration tools like Jenkins coming into the picture.

selenium testng jenkins - devops tools You can read more about Selenium and automation testing from this blog of mine: What is Selenium? Now, lets move onto the next topic in this DevOps tools blog.

3. Continuous Integration

This is the most brilliant DevOps phase. It might not make sense during the first cycle of release, but then you will understand this phase's importance going forward.

Wait, that is not completely correct. Continuous Integration (CI) plays a major role even during the first release. It helps massively to integrate the CI tools with configuration management tools for deployment.

Undisputedly, the most popular CI tool in the market is Jenkins. And personally, Jenkins is my favorite DevOps tool. Other popular CI tool are Bamboo and Hudson.

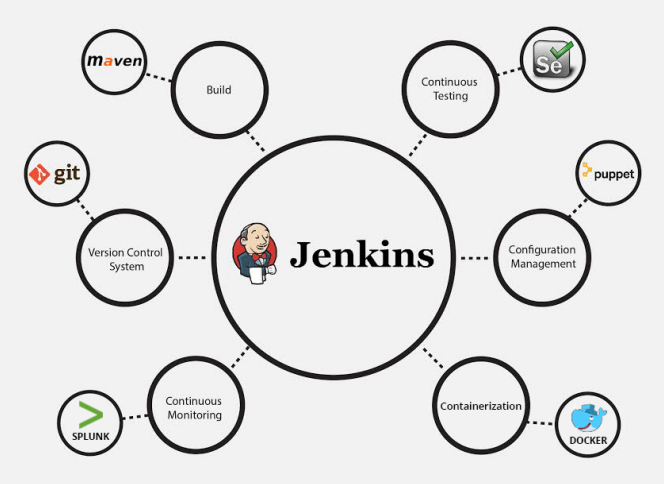
Why do I hold such a high regard for Continuous Integration tools? Because they are the one's which hold the entire 'DevOps structure' together.

It is the CI tools which orchestrates the automation of tools falling under other DevOps lifecycle phases. Be it, Continuous Development tools, or Continuous Testing tools, or Continuous Deployment tools, or even Continuous Monitoring tools, the Continuous Integration tools can be integrated with all of them.

When integrated with Git/ SVN, Jenkins can schedule jobs (pulling the code from shared repositories) automatically and make it ready for builds and testing (Continuous Development). Jenkins can build jobs either at scheduled times of day or when ever there is a commit pushed to the central repository. When integrated with testing tools like Selenium, we can achieve Continuous Testing. How? The developed code can be built using tools like Maven/ Ant/ Gradle. When the code is built, then Selenium can automate the execution of that code. How does it automate it? By creating a suite of test cases and executing the test cases one after the other.

The role of Jenkins/ Hudson/ Bamboo here would be to schedule/ automate “Selenium to automate test case execution”. When integrated with Continuous Deployment tools, Jenkins/ Hudson/ Bamboo can trigger the deployments planned by configuration management/ containerization tools. And finally, Jenkins/ Hudson can be integrated with monitoring tools like Splunk/ ELK/ Nagios/ NewRelic, to continuously monitor the status & performance of the server where the deployments have been made.

jenkins ci - devops tools



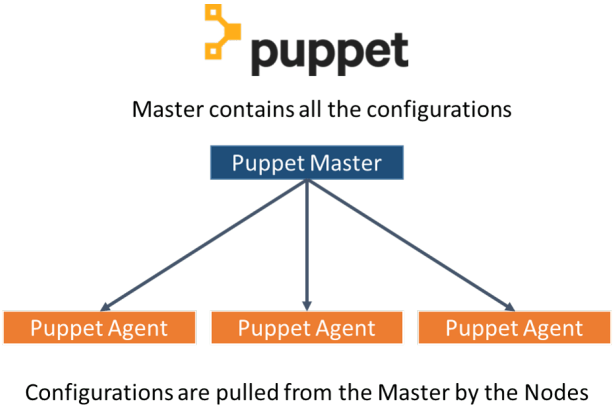
Because CI tools are capable of this and so much more, they are my favorite. Hence my statement: Jenkins is an elementary DevOps tool. You can read more about Jenkins here: What is Jenins?

4. Continuous Deployment

This (Continuous Deployment) is the phase where action actually happens. We have seen the tools which help us build the code from scratch and also those tools which help in testing. Now it is time to understand why DevOps will be incomplete without Configuration Management tools or Containerization tools. Both set of tools here help in achieving Continuous Deployment (CD).

Configuration Management Tools

Configuration Management is the act of establishing and maintaining consistency in an applications’ functional requirements and performance. In simpler words, it is the act of releasing deployments to servers, scheduling updates on all servers and most importantly keeping the configurations consistent across all the severs. For this, we have tools like Puppet, Chef, Ansible, SaltStack and more. But the best tool here is Puppet. Puppet & the other CM tools work based on the master-slave architecture. When there is a deployment made to the master, the master is responsible for replicating those changes across all the slaves, no matter the number! Amazing right?

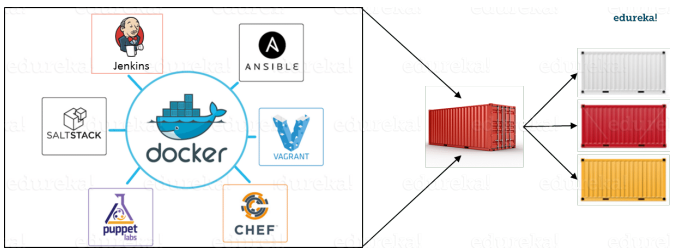


puppet architecture - devops tools

You can read more about Puppet here: What is Puppet? Now let's move onto Containerization.

Containerization Tools

Containerization tools are other set of tools which help in maintaining consistency across the environments where the application is developed, tested and deployed. It eliminates any chance of errors/ failure in production environment by packaging and replicating the same dependencies and packages used in development/ testing/ staging environment. The clear winner here is Docker, which was among the first containerization tool ever. Earlier, this act of maintaining consistency in environments was a challenge because VMs and servers were used, and their environments would have to be managed manually to achieve consitency. Docker containers threw this challenge up above and blew it out of the water. (Pun intended!)



Docker Integrations - What Is Docker Container

Another containerization tool is Vagrant. But off-late, a number of cloud solutions have started providing support for container services. Amazon ECS, Azure Container Service and Google Container Engine are a few of the cloud services that have started radical support for Docker containers. This is the reason why Docker is the clear winner. You can read more about Docker from here: What is Docker? So now, let's move on to the final topic in this DevOps tools blog.

5. Continuous Monitoring

Well, what is the point of developing an application and deploying it, if we do not monitor its performance. Monitoring is as important as developing the application because there will always be a chance of bugs which escape undetected during the testing phase.

Which tools fall under this phase? Splunk, ELK Stack, Nagios, Sensu, NewRelic are some of the popular tools for monitoring. When used in combination with Jenkins, we achieve Continuous Monitoring. So, how does monitoring help?

To minimize the consequences of buggy features, monitoring is a big add-on. Buggy features most often tend to cause financial loss. So, all the more reason to perform continuous monitoring. Monitoring tools also report failure/ unfavorable conditions before your clients/ customers get to experience the faulty features. Don't we all prefer this? Which is my favorite tool here? I would prefer either Splunk or ELK stack. These two tools are major competitors. They pretty much provide the same features. But the way they provide the functionality is where they are different.

Splunk

Splunk is a propriety tool (paid tool). But, this also effectively means that working on Splunk is very easy. ELK stack however, is a combination of 3 open-source tools: ElasticSearch, LogStash & Kibana. It maybe free, but setting it up is not as easy as a commercial tool like Splunk. You can try both of them to figure out the better for your organization. You can read more about Splunk here: What is Splunk?

What is DevOps Automation?

As we all aware Automation is the ultimate need for doing anything nowadays. In this, we will try to automate each and every step right from code generation till the code eventually gets pushed to code then monitoring it in real-time. In this article, I have divided most widely used DevOps tools based on different categories based on their usage.

Continuous Integration continuous deployment Tools

1. Jenkins

Perhaps the Most popular open-source continuous integration & continuous delivery servers. It allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.

2. Travis CI

Travis CI is a hosted continuous integration service used to build and test software projects hosted at GitHub and bitbucket. Open source projects may be tested at no charge via travis-ci.org. Private projects may be tested at travis-ci.com on a fee basis. TravisPro provides custom deployments of a proprietary version on the customer's own hardware.

3. Hudson

Hudson is a powerful and widely used open-source continuous integration server providing development teams with a reliable way to monitor changes in source control and trigger a variety of builds. Hudson excels at integrating with almost every tool you can think of. Use Apache Maven,

Apache Ant or Gradle or anything you can start with a command-line script for builds and send messages via email, SMS, IRC, and Skype for notifications.

4. TeamCity

It is a Java-based CI server package. TeamCity is a Server-based application that's very simple and easy to get familiar with and has an absolutely amazing browser-hosted dashboard. TeamCity also provides build progress, drill-down detail, and history information on the projects and configurations.

Containers*

1. Docker

Docker is a container management service. The keywords of Docker are developed, ship and run anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers that can then be deployed anywhere.

2. Kubernetes

It is an open-source container-orchestration system for automating application deployment, scaling, and management. It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation. It aims to provide a "platform for automating deployment, scaling, and operations of application containers across clusters of hosts".It works with a range of container tools, including Docker.

Configuration Management

1. Ansible

Ansible is an open-source IT Configuration Management, Deployment & Orchestration tool. It aims to provide large productivity gains to a wide variety of automation challenges. It helps to run tasks in sequence and create a chain of events that must happen on several different servers or devices.

2. Chef

Chef is a configuration management technology developed by Opscode to manage infrastructure on physical or virtual machines. It is an open-source developed using Ruby, which helps in managing complex infrastructure on the fly.

3. Puppet

Puppet is a configuration management tool developed by Puppet Labs in order to automate infrastructure management and configuration. Puppet is a very powerful tool which helps in the concept of Infrastructure as code. This tool is written in Ruby DSL language that helps in converting a complete infrastructure in code format, which can be easily managed and configured.

Infrastructure provisioning

1. Terraform

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions. Configuration files describe to Terraform the components needed to run a single application or your entire datacenter. Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

Infrastructure Monitoring

1. Nagios

Nagios is free to use open-source software tools for continuous monitoring. It helps you to monitor systems, networks, and infrastructure. It is used for continuous monitoring of systems, applications, service and business Processes in a DevOps culture.

Nagios runs plugins stored on the same server. It plugin's connects with a host or another server on your network or the Internet. Therefore, in the case of failure Nagios core can alert the technical staff about the issues. So, your technical team performs the recovery process before outage in the business processes.

Application monitoring

App Dynamics

AppDynamics is a leading Application Performance Management (APM) product. It is a tool that monitors your Application Infrastructure and gives you code-level visibility. It is supported for all major technologies (Java, .NET, PHP, Node.js, NoSQL, etc) and can be installed either as on-premise or as SaaS (Software As a Service) solution.

A piece of software called Agent is installed in the Application to be monitored. The Agent collects the performance metrics and sends them to a Server process called Controller. The controller processes the metrics and presents them via a Web Browser. A monitoring analyst can configure Alerts and generate reports using the Web Interface.

New Relic

New Relic's software analytics product for application performance monitoring (APM) delivers real-time and trending data about your web application's performance and the level of satisfaction that your end-users experience. With end to end transaction tracing and a variety of color-coded charts and reports, APM visualizes your data, down to the deepest code levels.

Logs Management

ELK Stack

The ELK Stack is a collection of three open-source products —Elasticsearch, Logstash, and Kibana. They are all developed, managed, and maintained by the company Elastic.

E stands for ElasticSearch: used for storing L stands for LogStash: used for both shipping as well as the processing and storing logs K stands for Kibana: is a visualization tool (a web interface) which is hosted through Nginx or Apache ELK Stack is designed to allow users to take to data from any source, in any format, and to search, analyze, and visualize that data in real-time.

Splunk

Splunk is used to search and analyze machine data. This machine data can come from web applications, sensors, devices or any data created by the user. It serves the needs of IT infrastructure by analyzing the logs generated in various processes, but it can also analyze any structured or semi-structured data with proper data modeling. It has built-in features to recognize the data types, field separators and optimize the search processes. It also provides data visualization on the search results.

Cloud Computing platforms

AWS Cloud

Amazon Web Services (AWS) is Amazon's cloud web hosting platform that offers flexible, reliable, scalable, easy-to-use, and cost-effective solutions. With this cloud, we need not plan for servers and other IT infrastructure which takes up much of time in advance. Instead, these services can instantly spin up hundreds or thousands of servers in minutes and deliver results faster. We pay only for what we use with no up-front expenses and no long-term commitments, which makes AWS cost-efficient.

Azure

It is a cloud computing platform, designed by Microsoft to successfully build, deploy, and manage applications and services through a global network of datacenters. Cloud computing can be referred to as the storing and accessing of data over the internet rather than your computer's hard drive. This means you don't access the data from either your computer's hard drive or over a dedicated computer network (home or office network)

Google Cloud

Google Cloud Platform is a suite of public cloud computing services offered by Google. The platform includes a range of hosted services for compute, storage and application development that run on Google hardware. Google Cloud Platform services can be accessed by software developers, cloud administrators and other enterprise IT professionals over the public internet or through a dedicated network connection.