

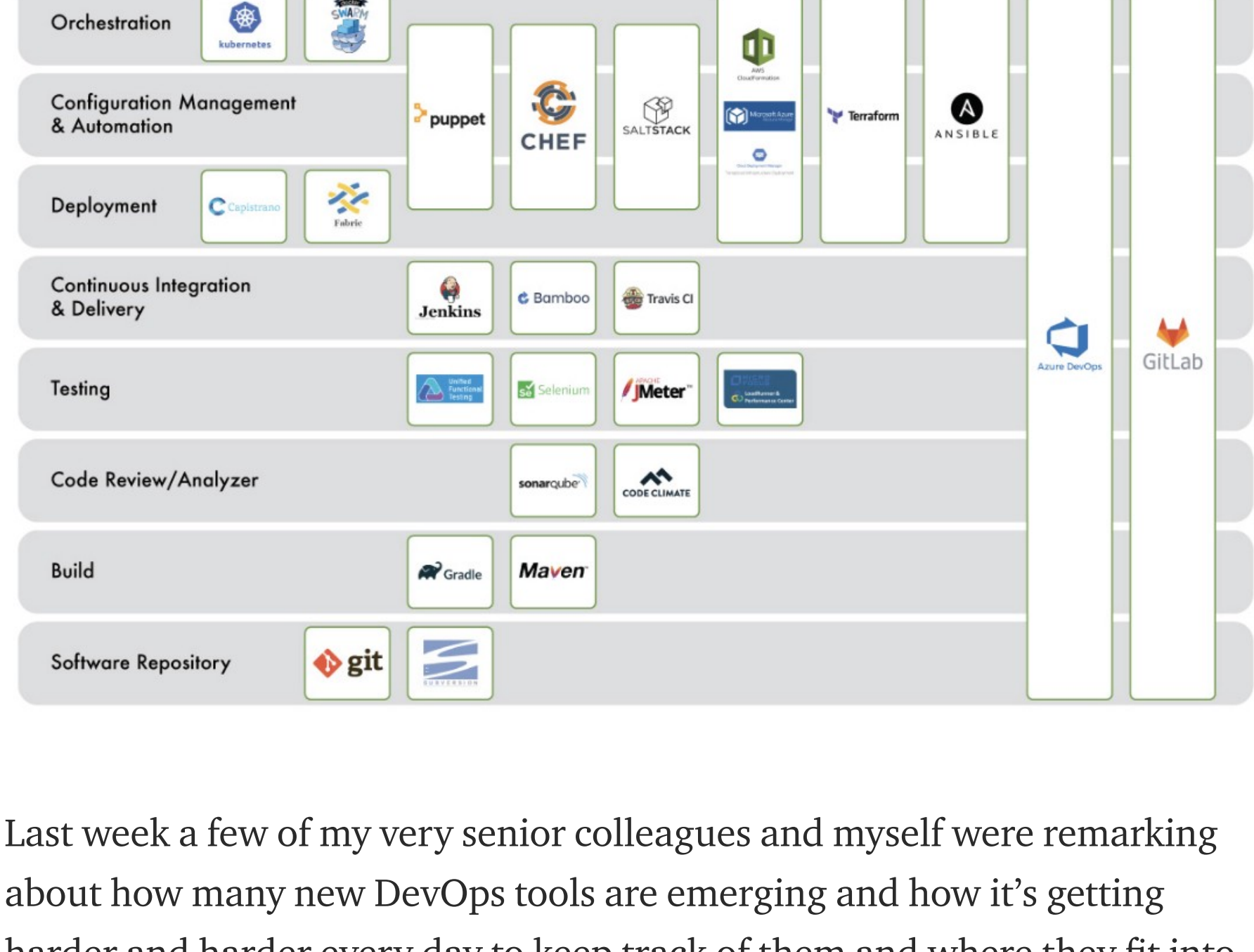
The 10-Minute Read to Understanding DevOps Tools



Joel Nylund [Follow](#)

Apr 26 · 7 min read ★



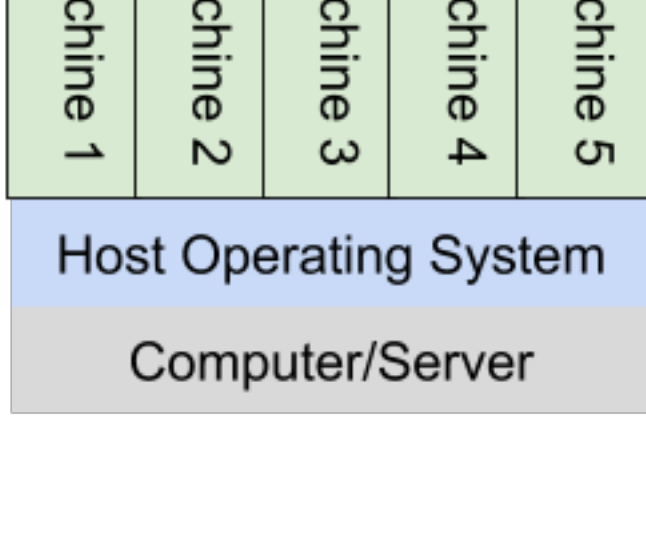


Last week a few of my very senior colleagues and myself were remarking about how many new DevOps tools are emerging and how it's getting harder and harder every day to keep track of them and where they fit into the world. I asked several of them where these tools, *Ansible*, *Terraform*, *Salt*, *Chef*, *Bamboo*, *CloudFormation*, fit in. Why would I use one vs. the other? Are they even the same thing? Am I missing a major player? I got back the same blank stares/questions that I had. So, I thought I would do some research, read, and try to make sense of it for all of us so we could classify products into categories or uses to which we are all familiar.

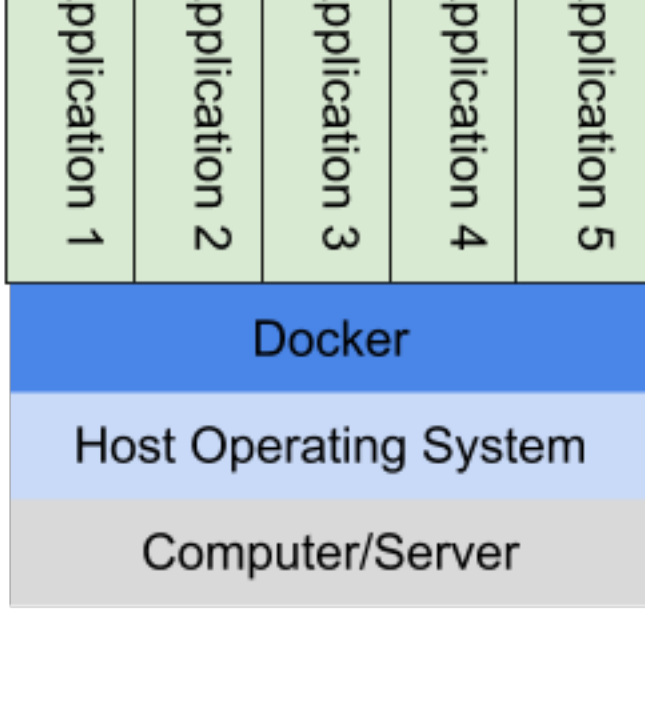
Before we start to talk about DevOps tools and categories, let's take a step back and discuss a few basic (but often overloaded) terms and what they mean.

Computer/Server — A physical device featuring a Central Processing Unit (CPU) and has memory (RAM), local storage (disk) and runs an operating system.

Virtual Machine — An emulation of a computer system running on a host computer; typically can be isolated from other operating systems in terms of CPU, memory and disk usage.



Containers — A packaging of software and all its dependencies so that it can run uniformly and consistently on any infrastructure. Docker containers are the most popular. They allow you to package up a bunch of stuff (your software, configurations and other software) for easy deployment and shipping. You can think of containers as the next evolution of virtualization (after Virtual Machines).



Network Device — A piece of hardware that routes network traffic between devices. Examples include routers, load balancers, and firewalls.

Software — Code that is written and runs on an operating system.

DevOps — Traditionally there was “development” (you build it), and there was “operations” (we will run it), and everything in between the two was subject to how a shop worked. Starting around 2010 and achieving near ubiquity around 2018, the DevOps idea is “a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.”

When you think about what goes into building and running a non-trivial system, there is actually a lot that goes into it. Here is a list of the traditional items to consider:

1. Obtaining the computer/server hardware
2. Configuring the computer/server hardware (operating systems, network wiring, etc.)
3. Monitoring the computer/server hardware
4. Obtaining the network devices (load balancers, firewalls, routers, etc.)
5. Configuring the network devices
6. Monitoring the network devices
7. Constructing the software
8. Building the software
9. Testing the software
10. Packaging the software
11. Deploying/releasing the software
12. Monitoring the software

Before DevOps, we used to have four distinct teams doing this work:

- Developer — they would perform #7, #8 and sometimes #10
- QA — they would perform #9 and sometimes #11
- System Administrator — they would do #1, #2, #3, #12
- Network Administrator — they would do #4, #5, #6

For the configuring of the hardware, network devices and software, each team likely would use their own set of scripts and tools and, in many cases, would do things manually to make a “software release” happen.

With the advent of DevOps, for me the key idea was breaking down these walls and making everyone part of “one” team, bringing consistency to the way all things are configured, deployed and managed.

Cloud — Defining the most overloaded term in the history of information technology is tough, but I like the t-shirt that says, “There is no cloud, it's just someone else's computer.” Initially, when the cloud services started they really were just someone else's computer (or a VM running on their computer), or storage. Over time, they have evolved into this and many, many value-added services. The hardware has been mostly abstracted away; you can't buy a hardware device in most cloud services these days, but you can buy a service provided by the hardware devices.

Infrastructure as Code (IAC) — A new ability or concept that allows us to define a complete setup of all the items in our data center including VMs, containers, and network devices through definition or configuration files. The idea is I can create some configurations and some scripts and run them using one of the tools we are about to discuss and they will automatically provision all of our services in the data center. CI/CD was a precursor to IAC, for years we have been working on automating our build/test/integration/deploy cycle, doing this with our cloud infrastructure was a natural extension to this. This brought about cost reduction, faster time to market, and less risk (of human error).

With the advent of IAC, many of the traditional development tools could now be used for managing infrastructure. Categories of tools (listed below) like software repositories, build tools, CI/CD, code analyzer, and testing tools that were traditionally used by software developers could now be used by DevOps engineers to build and maintain infrastructure.

“With the advent of DevOps, for me the key idea was . . . making everyone part of ‘one’ team, bringing consistency to the way all things are configured, deployed and managed.”

So now that we have some basic vocabulary defined, let me get back to the task of attempting to categorize DevOps tools in a way that makes it easier for us to determine what can be used for what.

1. **Software repository** — Tools to manage versions of software — Git is the most widely used today.
2. **Build tools** — Some software requires compiling before it can be packaged or used, traditional build tools include Make, Ant, Maven, and MSBuild.
3. **Continuous Integration tools** — Configured so each time you check code into a repository, it builds, deploys and tests the software. This usually improves quality and time to market. The most popular tools in this market are Jenkins, Travis, TeamCity and Bamboo.
4. **Code analyzer/review tools** — These tools look for errors in code, code formatting and quality, and test coverage. These vary from language to language. SonarQube is a popular tool in this space, as well as other “linting” tools.
5. **Configuration management** — Configuration management tools and databases typically store all the information about your hardware and software items as well as provide a scripting and/or templating system for automation of common tasks. There seems to be many players in this space. Traditional players were *Chef*, *Puppet* and *Salt*.
6. **Deployment tools** — These tools help with the deployment of software. Many CI tools are also CD (continuous deployment) tools which assist the deployment of software. Traditionally in Ruby, the Capistrano tool has been used widely; in Java, Maven is used by many. All of the orchestration tools also support some sort of deployment.
7. **Orchestration tools** — These tools configure, coordinate and manage computer systems and software. They typically include “automation” and “workflow” as part of their services. Kubernetes is a very popular orchestration tool that is focused on containers. Terraform is a very popular orchestration tool that is more broadly focused including cloud orchestration. Also, each cloud provider has their own set of tools (CloudFormation, GCP Deployment Manager, and ARM).
8. **Monitoring tools** — These tools allow the monitoring of hardware and software. Typically they include watchers that watch processes and log files to ensure the health of systems. Nagios is a popular monitoring tool.
9. **Testing tools** — Testing tools are used to manage tests, as well as test automation, including things like performance and load testing.

Of course, like any other set of products, the categories are not necessarily clean. Many tools cross categories and provide features from two or more categories. Below is my attempt to show most of the very popular tools and visualize where they fit in terms of these categories.

As you can see, there are several players like Ansible, Terraform and the cloud tools (AWS, GCP and Azure) that are trying to span the deployment, configuration management and orchestration categories with their offerings. The older toolset, Puppet, Chef and SaltStack, are focused on configuration management and automation but have expanded into orchestration and deployment. There are also tools like GitLab and Azure DevOps that are trying to span nearly every category of DevOps.

I hope this overview helps you understand the basics of DevOps, the categories of tools available, and how the various products on the market today help in one or more of these categories. At [Solution Street](#) we have used many of these tools over the years, for us there is no single “go to” tool we use in all cases. What is used is based on the technologies being used, where it is being hosted (and where in the future it may) as well as the talents and makeups of the team.


Further Reading:






[Blog Post: Why we use Terraform and not Chef, Puppet, Ansible, SaltStack, or CloudFormation](#)


[Blog Post: The Best Tools for Cloud Infrastructure Automation](#)

[AWS Website: DevOps — What is DevOps?](#)

Software Development | Technology | DevOps | Software Engineering | Tech

 171 claps






Joel Nylund

CEO at Solution Street — [www.solutionstreet.com](#). I have always loved solving problems and making things better. It is what drives me and it is my passion.

[Follow](#)



Level Up Coding

Coding tutorials and news. The developer homepage

[gitconnected.com](#)

[Follow](#)


See responses (1)

More From Medium

More from Level Up Coding

5 JavaScript Tips I Learned From Vue Source Code


bitfish in Level Up Coding
May 15 · 5 min read ★

 752

More from Level Up Coding

7 Things to Build When You Feel Bored as a Programmer


Daan in Level Up Coding
May 7 · 5 min read ★

 1.93K

More from Level Up Coding

How To Become A True Keyboard Warrior (And Stop Using Your Mouse)

keypressingmonkey in Level Up Coding
May 11 · 7 min read ★

 1.3K

Discover Medium

Welcome to a place where voices matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. [Watch](#)

Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. [Explore](#)

Explore your membership

Thank you for being a member of Medium. You get unlimited access to insightful stories from amazing thinkers and storytellers. [Browse](#)