theagileadmin.com

What Is DevOps?

DevOps is a term for a group of concepts that, while not all new, have catalyzed into a movement and are rapidly spreading throughout the technical community. Like any new and popular term, people have somewhat confused and sometimes contradictory impressions of what it is. Here's my take on how DevOps can be usefully defined; I propose this definition as a standard framework to more clearly discuss the various issues DevOps covers. Like "Quality" or "Agile," DevOps is a large enough concept that it requires some nuance to fully understand.

Definition of DevOps

DevOps is a new term emerging from the collision of two major related trends. The first was also called "agile system administration" or "agile operations"; it sprang from applying newer Agile and Lean approaches to operations work. The second is a much expanded understanding of the value of collaboration between development and operations staff throughout all stages of the development lifecycle when creating and operating a service, and how important operations has become in our increasingly service-oriented world (cf. Operations: The New Secret Sauce).

One definition Jez Humble explained to me is that DevOps is

"a cross-disciplinary community of practice dedicated to the study of building, evolving and operating rapidly-changing resilient systems at scale."

That's good and meaty, but it may be a little too esoteric and specific to Internet startup types. I believe that you can define DevOps more practically as

> **DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.**

A primary corollary to this is that part of the major change in practice from previous methods is

> **DevOps is also characterized by operations staff making use many of the same techniques as developers for their systems work.**

Those techniques can range from using source control to testing to participating in an Agile development process.

For this purpose, "DevOps" doesn't differentiate between different sysadmin sub-disciplines – "Ops" is a blanket term for systems engineers, system administrators, operations staff, release engineers, DBAs, network engineers, security professionals, and various other subdisciplines and job titles. "Dev" is used as shorthand for developers in particular, but really in practice it is

even wider and means “all the people involved in developing the product,” which can include Product, QA, and other kinds of disciplines.

DevOps has strong affinities with Agile and Lean approaches. The old view of operations tended towards the “Dev” side being the “makers” and the “Ops” side being the “people that deal with the creation after its birth” – the realization of the harm that has been done in the industry of those two being treated as siloed concerns is the core driver behind DevOps. In this way, DevOps can be interpreted as an outgrowth of Agile – agile software development prescribes close collaboration of customers, product management, developers, and (sometimes) QA to fill in the gaps and rapidly iterate towards a better product – DevOps says “yes, but service delivery and how the app and systems interact are a fundamental part of the value proposition to the client as well, and so the product team needs to include those concerns as a top level item.” From this perspective, DevOps is simply extending Agile principles beyond the boundaries of “the code” to the entire delivered service.

Definition In Depth

DevOps means a lot of different things to different people because the discussion around it covers a lot of ground. People talk about DevOps being “developer and operations collaboration,” or it’s “treating your code as infrastructure,” or it’s “using automation,” or “using kanban,” or “a toolchain approach,” or “culture,” or a variety of seemingly loosely related items. The best way to define it in depth is to use a parallel method to the definition of a similarly

complex term, agile development. Agile development, according to Wikipedia and the agile manifesto, consists of four different "levels" of concern. I've added a fifth, the tooling level – talk about agile and devops can get way too obsessed with tools, but pretending they don't exist is also unhelpful.

- **Agile Values** – Top level philosophy, usually agreed to be embodied in the Agile Manifesto. These are the core values that inform agile.

- **Agile Principles** – Generally agreed upon strategic approaches that support these values. The Agile Manifesto cites a dozen of these more specific principles. You don't have to buy into all of them to be Agile, but if you don't subscribe to many of them, you're probably doing something else.

- **Agile Methods** – More specific process implementations of the principles. XP, Scrum, your own homebrew process – this is where the philosophy gives way to operational playbooks of "how we intend to do this in real life." None of them are mandatory, just possible implementations.

- **Agile Practices** – highly specific tactical techniques that tend to be used in conjunction with agile implementations. None are required to be agile but many agile implementations have seen value from adopting them. Standups, planning poker, backlogs, CI, all the specific artifacts a developer uses to perform their work.

- **Agile Tools** – Specific technical implementations of these

practices used by teams to facilitate doing their work according to these methods. JIRA Agile (aka Greenhopper), planningpoker.com, et al.

Ideally the higher levels inform the lower levels – people or organizations that pick up specific tools and practices without understanding the fundamentals may or may not see benefits but this "cargo cult" approach is generally considered to have suboptimal results. I believe the different parts of DevOps that people are talking about map directly to these same levels.

- **DevOps Values** – I believe the fundamental DevOps values are effectively captured in the Agile Manifesto – with perhaps one slight emendation to focus on the overall service or software fully delivered to the customer instead of simply "working software." Some previous definitions of DevOps, like Alex Honor's "People over Process over Tools," echo basic Agile Manifesto statements and urge dev+ops collaboration.

- **DevOps Principles** – There is not a single agreed upon list, but there are several widely accepted attempts – here's John Willis coining "CAMS" and here's James Turnbull giving his own definition at this level. "Infrastructure as code" is a commonly cited DevOps principle. I've made a cut at "DevOps'ing" the existing Agile manifesto and principles here. I personally believe that DevOps at the conceptual level is mainly just the widening of Agile's principles to include systems and operations instead of stopping its concerns at code checkin.

- **DevOps Methods** – Some of the methods here are the same; you can use Scrum with operations, Kanban with operations, etc. (although usually with more focus on integrating ops with dev, QA, and product in the product teams). There are some more distinct methods, like Visible Ops-style change control and using the Incident Command System for incident reponse. The set of these methodologies are growing; a more thoughtful approach to monitoring is an area where common methodologies haven't been well defined, for example.

- **DevOps Practices** –Specific techniques used as part of implementing the above concepts and processes. Continuous integration and continuous deployment, "Give your developers a pager and put them on call," using configuration management, metrics and monitoring schemes, a toolchain approach to tooling… Even using virtualization and cloud computing is a common practice used to accelerate change in the modern infrastructure world.

- **DevOps Tools** – Tools you'd use in the commission of these principles. In the DevOps world there's been an explosion of tools in release (jenkins, travis, teamcity), configuration management (puppet, chef, ansible, cfengine), orchestration (zookeeper, noah, mesos), monitoring, virtualization and containerization (AWS, OpenStack, vagrant, docker) and many more. While, as with Agile, it's incorrect to say a tool is "a DevOps tool" in the sense that it will magically bring you DevOps, there are certainly specific tools being developed with the express goal of facilitating the above

principles, methods, and practices, and a holistic understanding of DevOps should incorporate this layer.

In the end, DevOps is a little tricky to define, just like its older brother Agile. But it's worth doing. When left at the pure philosophy level, both can seem like empty mom-and-apple-pie statements, subject to the criticism "You're just telling me 'do my job better,' duh…" But conversely, just the practices without the higher level guidance turn into a cargo cult. "I do what this Scrum book says so I'm doing Agile" is as erroneous as "I'm using Chef so I'm DevOps right?" To be a successful Agile or DevOps practitioner is to understand all the layers that go into it, and what a given DevOps implementation might contain or not contain. In the end, what DevOps hopes to bring to Agile is the understanding and practice that software isn't done until it's successfully delivered to a user and meets their expectations around availability, performance, and pace of change.

History of DevOps

The genesis of DevOps comes from an increasing need for innovation on the systems side of technology work. The DevOps movement inherits from the Agile System Administration movement and the Enterprise Systems Management (ESM) movement.

ESM, which arose in the mid-2000's, provided the original impetus of "Hey, our methodology of running systems seems to still be in a pretty primitive state despite years of effort. Let's start talking about doing it better." John Willis, whurley, and Mark Hinkle from Zenoss

were involved in that, and sponsored a BarCamp around the concept. I think during this phase, initial enchantment with ITIL as a governance framework was largely overthrown for the "ITIL Lite" Visible Ops approach, as well as a shift from being "large vendor" focused – used to be, the enterprise frameworks like HP, IBM, and CA were the only meaningful solutions to end to end systems management, but more open source and smaller vendor stuff was coming out, including Spiceworks, Hyperic, Zenoss, and others.

Also in 2008, the first Velocity conference was held by O'Reilly, focusing on Web performance and operations, which provided a venue for information sharing around operations best practices. In 2009 there were some important presentations about the developer/operations collaboration at large shops (most notably Flickr) and how that promoted safe, rapid change in Web environments. Provisioning tools like Puppet and Chef had strong showings there. More people began to think about these newer concepts and wonder how they might implement them.

Somewhat in parallel, as agile development's growth in the development space was reaching its most fevered pitch and moving from niche to common practice, this turned into thinking about "Agile Systems Administration" especially in Europe. Gordon Banner of the UK talked about it early on with this presentation. A lot of the focus of this movement was on process and the analogies from kanban and lean manufacturing processes to IT systems administration. Then in 2009, Patrick Debois from Belgium and Andrew "Clay" Shafer from the US met and started talking up (and coined the term) DevOps, and then Patrick held the

first DevOpsDays event in Ghent that lit the fuse. The concept, now that it had a name, started to be talked up more in other venues (I found out about it at OpsCamp Austin) including Velocity and DevOpsDays here in the US and spread quickly.

In Patrick Debois' view, DevOps arose as a reaction against the silos and inflexibility that were resulting from existing practices, which probably sounds familiar. Here's a good piece by John Willis on the history of the DevOps movement that deconstructs the threads that came together to create it.

DevOps emerged from a "perfect storm" of these things coming together. The growing automation and toolchain approach fed by more good monitoring and provisioning tools, the need for agile processes and dev/ops collaboration along with the failure of big/heavy implementations of ITSM/ITIL – they collided and unconsciously brought together all three layers of what you need for the agile movement (principles, process, and practices) and caught fire. Since then it has developed further, most notably by the inclusion of Lean principles by many of the thought leaders.

What is DevOps Not?

It's Not NoOps

It is not "they're taking our jobs!" Some folks thinks that DevOps means that developers are taking over operations and doing it themselves. Part of that is true and part of it isn't.

It's a misconception that DevOps is coming from the development

side of the house to wipe out operations – DevOps, and its antecedents in agile operations, are being initiated out of operations teams more often than not. This is because operations folks (and I speak for myself here as well) have realized that our existing principles, processes, and practices have not kept pace with what's needed for success. As businesses and development teams need more agility as the business climate becomes more fast paced, we've often been providing less as we try to solve our problems with more rigidity, and we need a fundamental reorientation to be able to provide systems infrastructure in an effective manner.

Now, as we realize some parts of operations need to be automated, that means that either we ops people do some automation development, or developers are writing "operations" code, or both. That is scary to some but is part of the value of the overall collaborative approach. All the successful teams I've run using this approach have both people with deep dev skill sets *and* deep ops skill sets working together to create a better overall product. And I have yet to see anyone automate themselves out of a job in high tech – as lower level concerns become more automated, technically skilled staff start solving the higher value problems up one level.

It's Not (Just) Tools

DevOps is also not simply implementing a set of tools. One reason why I feel that a more commonly accepted definition of DevOps is needed is that having various confusing and poorly structured definitions increases the risk that people will pass by the "theory" and implement the processes or tools of DevOps without

the principles in mind, which is definitely an antipattern. Automation is just the exercise of power, and unwise automation can do as much damage as wise automation can bring benefit.

Similarly, Agile practitioners would tell you that just starting to work in iterations or adopting other specific practices without initiating meaningful collaboration is likely to not work out real well. There are some teams at companies I've worked for that adopted some of the methods and/or tools of agile but not its principles, and the results were suboptimal. Sure, a tool can be useful in Agile (or DevOps), but if you don't know how to use it then it's like giving an assault weapon to an untrained person.

But in the end, fretting about "tools shouldn't be called DevOps" is misplaced. Is poker planning "agile" in the sense that doing it magically gets you Agile? No. But it is a common tool used in various agile methodologies, so calling it an "agile tool" is appropriate. Similarly, just because DevOps is not just a sum of the tools doesn't mean that tools specifically designed to run systems in accordance with a DevOps mindset aren't valuable. (There are certainly a bunch of tools I've used that seem specifically designed to prevent it!)

It's Not (Just) Culture

Many people insist that DevOps "is just culture" and you can't apply the word to a given principle or practice, but I feel like this is overblown and incorrect. Agile has not helped thousands of dev shops because the work on it stopped at "culture," with admonitions

to hug coworkers and the lead practitioners that identified the best practices simply declaring it was all self-evident and refusing to be any more prescriptive. (Though there is some of that). DevOps consists of items at all the levels I list above, and is largely useless without the tangible body of practice that has emerged around it. You might be able to figure out all those best practices yourself given the cultural direction and lots of time to experiment, but sharing information is why we have the Internet (and printing press for that matter).

It's Not (Just) Devs and Ops

And in the end, it's not exclusionary. Some people have complained "What about security people! And network admins! Why leave us out!?!" The point is that all the participants in creating a product or system should collaborate from the beginning – business folks of various stripes, developers of various stripes, and operations folks of various stripes, and all this includes security, network, and whoever else. There's a lot of different kinds of business and developer stakeholders as well; just because everyone doesn't get a specific call-out ("Don't forget the icon designers!") doesn't mean that they aren't included. The original agile development guys were mostly thinking about "biz + dev" collaboration, and DevOps is pointing out issues and solutions around "dev + ops" collaboration, but the mature result of all this is "everyone collaborating". In that sense, DevOps is just a major step for one discipline to join in on the overall culture of agile collaboration that should involve all disciplines in an organization. So whoever is participating in the delivery of the software or service

is part of DevOps.

It's Not (Just) A Job Title

Simply taking an existing ops team and calling them "The DevOps Team" doesn't actually help anything by itself. Nor does changing a job title to "DevOps Engineer." If you don't adopt the values and principles above, which require change at an overall system level not simply within a given team, you won't get all the benefits.

However, I'm not in the camp that rails that you 'can't have DevOps in a job title." It is often used in a job title as a way to distinguish "new style DevOps-thinking, automation-first, dev-collaborating, CI-running, etc. sysadmin" from "grouchy back room person who aggressively doesn't care what your company does for a living." Some people find value in that, others don't, and that's fine. As a hiring manager myself, I see a clear difference in the fit of applicants when I put it on a job posting for a systems engineer, which provides an incentive for me to keep doing so…

It's Not Everything

Sometimes, DevOps people get carried away and make grandiose claims that DevOps is about "everything everywhere!" Since DevOps plugs into the overall structure of a lot of lean and agile thinking, and there are opportunities for that kind of collaboration throughout an organization, it's nice to see all the parallels, but going and reengineering your business processes isn't really DevOps per se. It is part of an overall, hopefully collaborative and

agile corporate culture, but DevOps is specifically about how operations plugs into that. Some folks overreach and end up turning DevOps into a super watered down version of Lean, Agile, or just love for everyone. Which is great at the vision level, but as you march down the hierarchy of granularity, you end up mostly dealing with operational integration – other efforts are worrying about the other parts (you can personally too of course). But there are still a lot of unsolved problems around the delivery of software and maintenance of services and making it fast, reliable, secure, et al. – if someone wants to use what they've learned from DevOps to go be a larger scope corporate consultant that's fine, but most people involved in DevOps are technical practitioners who are looking for better ways to do *their* job, not someone else's. In Agile there is "Agile Software Development" and then there's the larger Agile organization work. I think DevOps is best defined as "Agile Software Delivery and Operations," which should similarly work in concert with others working on larger organizational initiatives, but without losing sight of its primary value proposition for the organization.

Getting Started With DevOps

There's not one path to DevOps – there's just what works in your organization. Very successful DevOps initiatives have been originated from dev teams and from ops teams, top down and bottom up, from inside the company and from consultants, with widespread education and with skunkwork pilots. Therefore it's hard to give a generic playbook for how you can get it implemented. I think it's safe to say that it starts with you yourself learning about

the values, principles, methods, and practices of DevOps and trying to spread it via whatever channel is most effective – telling fellow techies, getting management buyin, just starting to implement things in a more DevOps way yourself and letting success speak for itself… People will try to tell you how things can rise to success in your org, but that advice is usually more policy and wishful thinking than reality. Observe how other popular things in your organization have arisen and gained currency and try those same channels. And keep learning.

DevOps Reading List

DevOps is still new so an undefined batch of blogs that changes monthly and following people on Twitter is often the best source of up to date information. Yes, that's annoying. However, there are several books and other reliable sources of good information you can use and then share with others. Ironically while there are a few books with "DevOps" in the title I don't recommend any of them.

- The Phoenix Project, Gene Kim, George Spafford, Kevin Behr – In novel format inspired by the seminal Lean work The Goal, this is a narrative of a DevOps implementation in a troubled software company.

- Web Operations, various – An O'Reilly book collecting a series of essays on Web operations that are really thoughts from a lot of the key DevOps pioneers.

- Continuous Delivery, Jez Humble and David Farley – While CI/CD

isn't the sum total of DevOps like some people would have it, it's certainly a major area of innovation and this is the definitive work on it.

- A Practical Approach to Large-Scale Agile Development, Gary Gruver – For those who think DevOps is just for startups or just for Web software, this is the tale of how the HP LaserJet firmware division transitioned to an agile/CI/DevOps structure.

- The Practice of Cloud System Administration, Tom Limoncelli, Strata Chalup, Christina Hogan – A textbook style guide from the operations side, with loads of great new-style systems guidance and a lot of explicit DevOps content.

- Release It!, Michael Nygard – There needs to be more books like this, it explains common systems failure patterns and success patterns – I think of it as the Gang of Four Design Patterns book for systems.

- Lean Software Development, Mary and Tom Poppendieck – Lean is being increasingly adopted within the DevOps community, but starting from Deming and TPS is somewhat intimidating. This book is the seminal work on Lean in software.

- And round it off with Gareth Rushgrove's DevOps Weekly email newsletter.