verifa

Continuous Everything

DevOps Manifesto

ABOUT VERIFA

Verifa has been helping leading technology organisations adopt DevOps methodologies since 2015, through a global network of consultants and offices.

We embrace DevOps and empower software development teams to deliver software faster, at higher quality and lower cost, than ever before by combining tooling, professional services and lots of automation.

Accelerating & Improving Project Velocity

We help companies in delivering high quality, reliable and secure software systems in an agile and compliant manner. Our experienced team enables continuous improvement across the entire software development life-cycle.

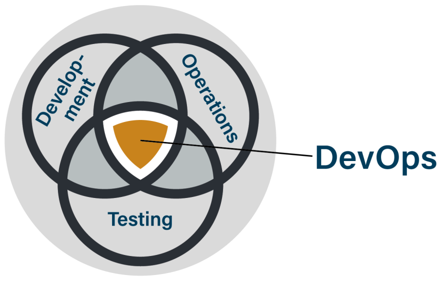
At Verifa we work with several core DevOps themes: values, principles, methods, practices, and tools. This document serves as an introduction to these concepts. Whether you have implemented DevOps and need support, or you are about to begin a revolution, Verifa, an industry-leading organization, can help you stay ahead of the curve.

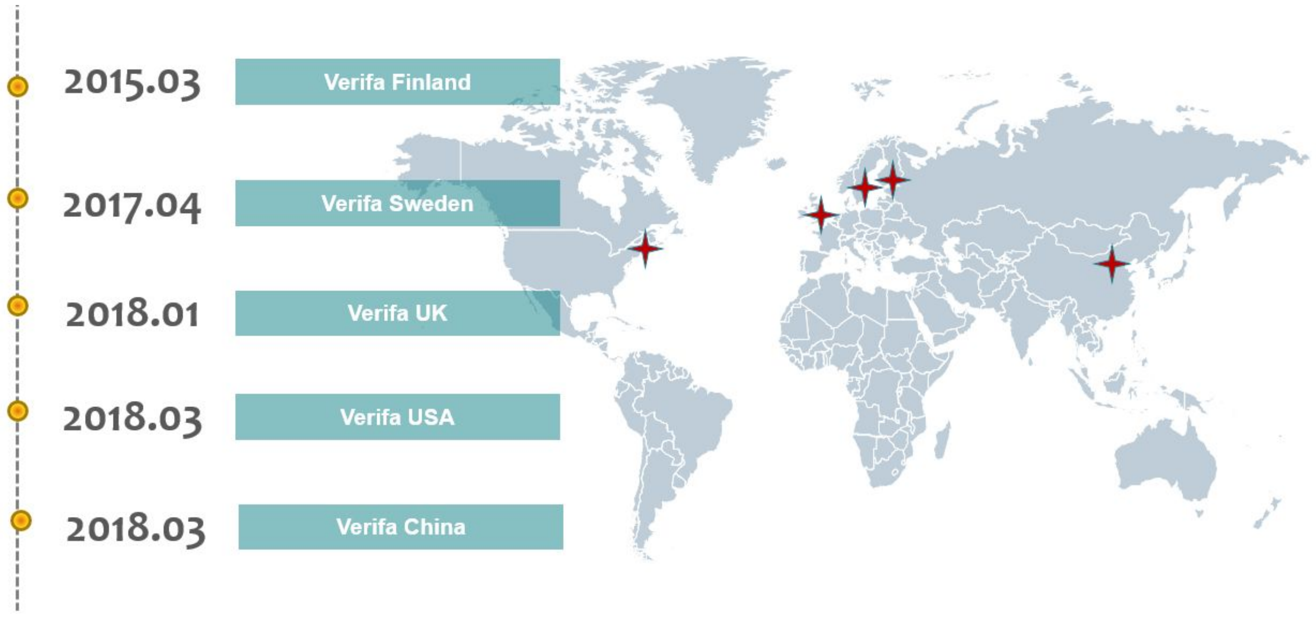
ABOUT DEVOPS

The concepts outlined here will help you on the path to realising the benefits of agile, robust and efficient software delivery. DevOps is primarily about cultural change. It provides the structure and narrative to improve any stage of your software development and delivery process.

Agile, Robust, Efficient

The method behind DevOps can improve IT processes and business outcomes. And it makes your daily life easier. By combining agile software development and deployment with robust application quality, results in a reduction in cost associated with fixing problems and deploying new features. And you gain vast improvements in overall business and IT efficiency.





VALUES

"We aim to do this by being known as the world's most professional and dedicated company committed to improving software quality. We embrace the open source ethos, and enthusiastically contribute our expertise for the betterment of open source code that benefits the public."

- Neil Langmead, CEO and President of Verifa

Businesses have been delivering software-based solutions for decades, but success and satisfaction rates with IT continue to be low. The Standish Group Chaos Report[1] found that only 29% of IT project implementations are successful, and it found 19% complete failures. These are projects that have gone over budget, the timescales have slipped or the project was completely abandoned before completion. Breaking out of the cycle of poor IT requires a shift on the human side of your organisation. While we often think DevOps is a technology problem, in reality, it's a cultural and business issue. The first hurdle is to impress on your team the importance of key values driving the movement: Culture, Automation, Measurement, and Sharing (CAMS). These values are the "why" behind the adoption of a DevOps culture.

Culture drives behaviour. It exists when people have a mutual understanding of each other and what they are trying to achieve. Early in IT organizations, software managers adopted the structure of engineering companies, the type that builds bridges and cars. They split teams into distinct groups: Development, created features; Test was next, to check the features against a documented specification; Operations are then tasked with maintaining stability and; Support deal with end users. Walls formed around these silos in response to the teams differing goals. Now we find these groups don't speak the same language. There is little mutual understanding.

By breaking down the barriers and integrating your team around a common set of goals you can realise performance gains. Changing underlying behaviours is how to drive change in company culture.

Building Better Software

Our dream to is make software developers, project managers and those responsible for delivering software extremely happy. To empower software development teams to deliver software faster, at higher quality and lower cost than ever before. This can be achieved by following a journey of DevOps based tooling, professional support and automated pipeline based processes.

PRINCIPLES

Like all efficient processes the underlying DevOps principles are simple. Adopt systems thinking, amplify feedback loops, and instil a culture of continuous experimentation and learning. Through better communication and a more open culture, it's easy to deliver faster product release and higher quality code using tooling techniques.

In a traditional environment, the development teams are tasked with developing new code, fixing compilation bugs and churning out 'product' as fast as possible. Operations teams are responsible for maintaining stability and controlling change. Its primarily the separation of duties which causes a conflict of interest and destroys communication between the teams.

[1] https://www.information-age.com/projects-continue-fail-alarming-rate-123470803/

“Adopting DevOps is an amazing way to transform your business . It improves business agility and customer experience. But most importantly it makes coding fun again!
It involves a cultural shift in all team members from the work experience guy, all the way up to senior management.”

- Adam Mackay, Principal Consultant at Verifa

As a DevOps pioneer, it is your responsibility to apply methods to break out of inefficient silos.

METHODS

"Trust, but verify" - Ronald Reagan

Lean Management

One of the constants when working in technology is speed of change. Lean management principles allow business to embrace change and are one of the cornerstones in instilling a DevOps culture. Lean is all about efficiently running a business, and getting products to your customers quickly.

The highest priority for a Lean practitioner is to understand your customers problems and build out from there. In DevOps the goal is getting feedback from customers quickly, through incremental deliveries which feed the cycle of improvements. Features to pursue should be selected based on good, robust data.

Rewarding team members for the amount of output created is very effective when the work is essentially algorithmic. ie the work follows a pre-approved plan. However, this is not effective in modern knowledge based work where humans have to solve problems as well as perform tasks. Lean is the way to manage people doing more complex work. Generally the aim of supervision should be to help people do a better job rather than micro manage each activity.

Teams and managers need to learn how to plan, organise and work in an atmosphere of collective responsibility.

In our experience the approach correlates well with organizational performance. You have probably heard of the term ‘minimum viable process’. Applying this in a DevOps context means not to add process unless the whole team agrees that is really the solution. It is vital to continually prune unnecessary processes to achieve operational efficiency.

The more trust and discretion that you can give to the people doing the work, the better results you'll get. Simply following these steps will get you most of the way:

- Reorganize into independent cross-functional teams.
- Apply lean agile processes to keep your effort focused on the creation of value.
- Address the fears people have that are barriers to change.

People over Process over Tools

It takes a wide range of skills to create and deliver a market leading product. Skills are possessed by people and not tools, so it is logical to make people the priority; they make up the fabric of your organisation.

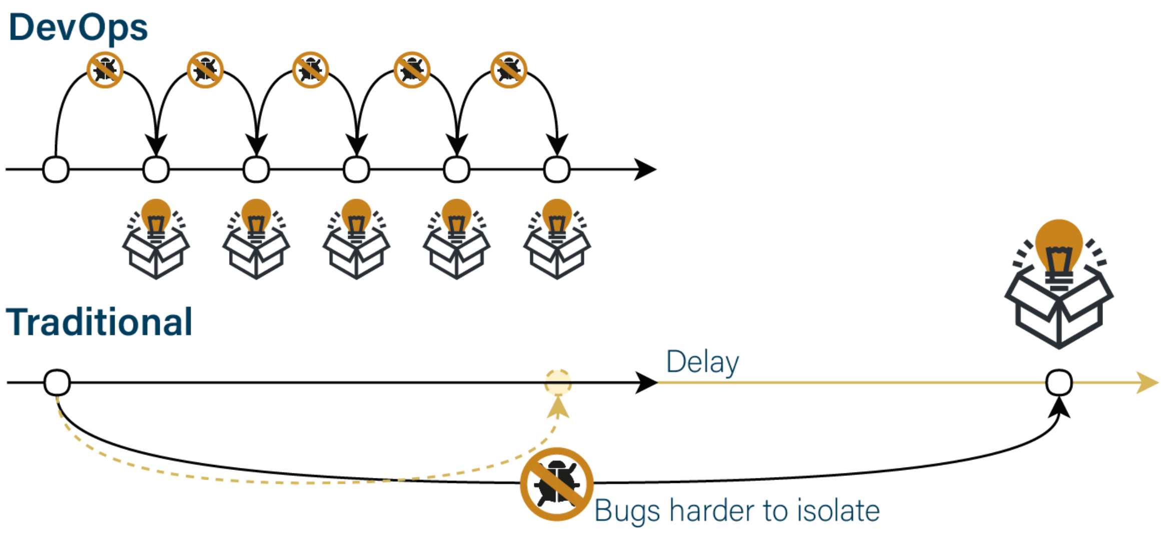
Once your team has devised and prioritized a portfolio of features that will form the basis of a delivery increment, you need to have confidence that the team can work on autopilot as the work is enacted. The development of new ideas is encouraged in DevOps. And ideas can come from anyone on the team. Historically, in the technology domain, teams mistakenly operate under the assumption that ideas are the responsibility of the senior members alone. In reality, junior members often have the best perspectives on the pain points of customers.

As new ideas emerge mid-project, a strong management process is important to prioritize and manage the work to be done. There's no reason why new ideas can't be added to the your project throughout the cycle, but it is important to employ objective criteria, such as strategic linkage, cost benefit analysis, project interdependency, and risks, to provide a realistic picture as to the ranking of any new feature. This will help avoid scenarios where a powerful and loud team member constantly pushes to reshuffle priorities.

Once your culture has shifted towards DevOps thinking, you can create automation and bring in tools that allow you to control your systems. Automation is that accelerator that's going to help you achieve all the other benefits of DevOps.

CD is the additional practice of deploying every change to a production-like environment and performing automated integration and acceptance testing after it passes its build and unit tests. This can involve small-scale deployment on a single server using test environments created with Docker containers or virtual machines to mimic a production environment, enabling you to run realistic integration tests.

Testing should end with deployment to a production-like environment. This allows validation of the deployment process and checks on the functionality and performance in real life.



Continuously Deliver, Incrementally

Continuous Delivery

When implementing the traditional model for delivering software, most of the time is spent in development of the application and very little time is spent with the application running, at least in a complete state. The old way of working is to build and deploy the application, and submit it for a test phase at the end. This results in bug reports arriving in large batches, late in the project.

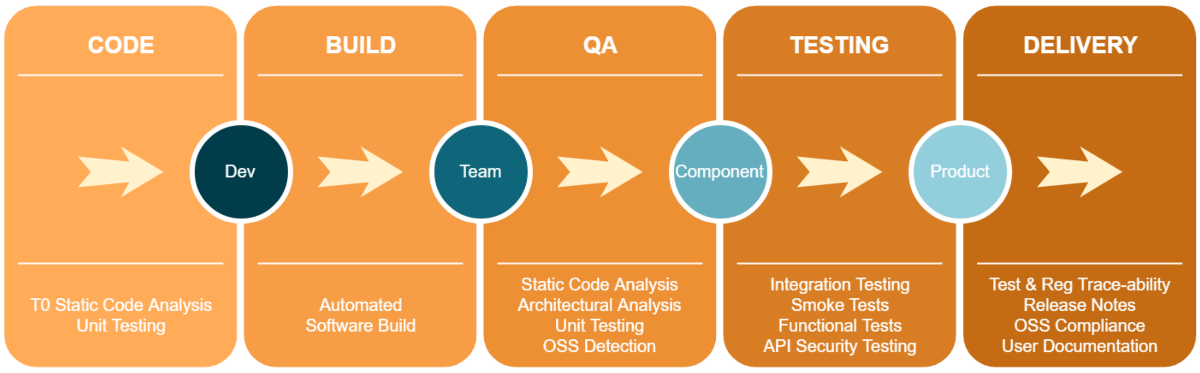
With DevOps we deploy what we call Continuous Integration (CI) and Continuous Delivery (CD).

CI is the practice of automatically building and unit testing an entire application frequently, ideally on every source code check-in, many times a day if necessary. The aim is not to use long-running parallel code branches that require a messy merge. Or if it is unavoidable, at least set up builds for the development branches to continually validate that they work.

A continuous delivery pipeline should contain all of the following in an automated or scripted process:

- Build
- Static Code Analysis
- Architectural Analysis
- Unit tests
- Integration tests
- Functional tests
- Security tests
- Deployment to a production like environment

Continuous delivery is all about coding, testing, and releasing software frequently, in really small



batches so that you can improve overall quality and velocity.

Change Control

"Progress is impossible without change."

- George Bernard Shaw

DevOps embraces the ability to constantly change. But change does not just happen, it must be managed, and establishing a Lean Culture must be managed from the very top of the organization. Change control is a broad term used to describe the processes and tools used to redirect people and resources.

There is a direct correlation between operational success and control over changes. This doesn't mean you need a traditional control process which slows you down and can do more harm than good. The requirement here is a scalable, resilient continuous delivery platform. Focus on eliminating fragile artefacts, creating a repeatable build process, managing dependencies and creating an environment of continual improvement.

"To improve is to change; to be perfect is to change often."

- Winston Churchill

For a DevOps enabled organization, these are certainly words to live by.

Infrastructure as Code (Configuration as Code… Everything as Code)

In our experience a large number of operations teams don't use source control at all. Let alone for their scripts or supporting processes. The ideal is to treat systems like code. Develop them in your code editor, run unit and integration tests as part of a continuous integration pipeline, and deploy them.

That's what infrastructure as code is about.

If you're new to Cloud infrastructure it can look like a lot of effort. Traditional systems engineers are familiar with building a server and crafting its operating system and configuration. This doesn't scale in the cloud however. The cloud is just other people's computers, managed by people who are much better at cable management than you. The cloud provides the elasticity you can't get in your own data-centre. You can scale to thousands of CPUs or exabytes of data if needed; it's impossible to set that up by hand.

By maintaining your entire system as code you achieve three key benefits:

Auditable. Could manually configured infrastructure pass an audit? Yes, if it is well documented and tested. But it's far easier and cheaper to maintain auditability on virtualized infrastructure.

Repeatable. You may be familiar with setting up a stack in the cloud. It's not unheard of to click through 40 or so pages of AWS web UI to build your development stack. And once it's running how long is it going to take to repeat the process for a staged release? And again for deployment? With scripted virtualisation this set-up process can and should be managed as code.

Dependable. If it's difficult to audit and repeat the actions that built your infrastructure reliably, you can't depend on them.

Once you enter 'everything as code' mindset you suddenly get all the benefits software developers have… source code control, collaboration tools, automated testing and deployment.

PRACTICES

There are many specific practices that we have found to help organisations in their DevOps journey. A few of the main ones are detailed below, but there are many more.

Incident Command System, is a method for dealing with major incidents with self-organizing teams who quickly and effectively coordinate efforts and scale up and down and evolve as the incident changes in scope, scale, and focus.

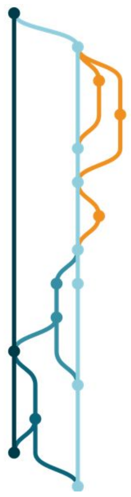
Devs in Loop. DevOps practice has the greatest impact on application management, service desk, and the IT operations function. You put your developers directly responsible for the service they create to achieve a fast feedback loop that helps improve logging and deployment; and enables faster resolution of issues. You will find you can respond much faster not just to bugs, but also to new features and ensure the right product is being developed.

Transparent Reporting, simply means easily-accessible communication during outages to increase user satisfaction. Produce visible status pages for each of your services. These should let users know what's happening, what's being done about it, and, after the issue is resolved, what steps you are taking to avoid reoccurance. It's important to sound like a human when dealing with users.

Blameless Postmortems to engender a culture of blamelessness. If your culture is one of kill the messenger or the nail that sticks up gets hammered down, you've got to face that down and address what behaviors get rewarded in your organization. If you want to adopt blameless postmortem practices for the analysis of major incidents, problems, and IT service continuity or disaster recovery events, it's best to start with a better blameless agenda to enforce behavior.

T(X) : Isolate-verify-merge. At Verifa we thrive on providing the maximum quality in all that we do. Out of this drive we have developed our own approach to minimise wastage and maximise quality. We call it T(X).

When a developer introduces technical debt (e.g. failing build, new architecture or static analysis issues, failing unit test cases) and merges this with a shared development branch, all members working on that development branch will now inherit this technical debt. T(x) provides checkpoints in the



master branch is where deliveries are made from
dev branch is the shared development branch
feature branches are created from the dev branch
release branch is used for release candidates
support branch is used for hotfixes required once a release has been made

Following this definition of branching, we can define T(x) as follows:
T(1): feature branches
T(2): dev branch
T(3): release branch
T(4): support branch
T(5): master branch

T(X) is a novel time-based approach developed by Verifa to minimize technical debt.

development process to avoid
the accumulation of technical debt:

- T0 daily at the desktop
- T1 incrementally on feature branches
- T2 incrementally on integration branches
- T3 scheduled for long-lived activities

By selecting and deploying the right practices you can help ensure the success of your DevOps adoption.

TOOLS

DevOps has caused a recent explosion in the number of tools available such as source code management, database management, build, continuous integration, testing, deployment, containerization, cloud, release management, collaboration, business intelligence, monitoring, logging, security...

The most successful tools are point solutions that can be interconnected. This is referred to as your DevOps tool chain. There are three characteristics that a successful DevOps tool chain should exhibit.

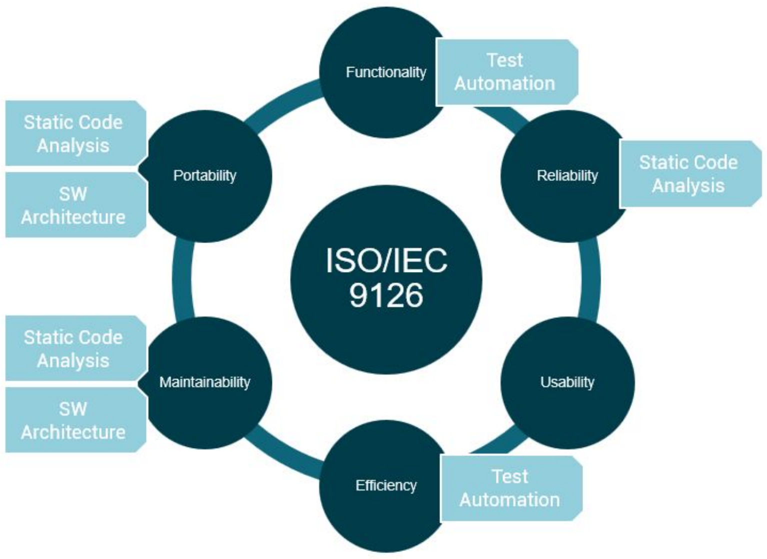
Programmable. You should be able to call and invoke the tool chain through an API or on the command line. Generally UI driven tools detract from the automated nature of DevOps and slow the process down.
Verifiable. That means, it exposes clearly what it's doing and provide some manner of validating that it did what it was supposed to do correctly.
Well-behaved. It is important to select tools that integrate well into a complete, automated toolchain.

Tools are important as they allow your people to spend more of their time implementing features and less time on repetitive tasks. But a tool is only useful to the degree that it supports your entire system.

At Verifa we have implemented a number of automated tool chains for our clients based on leading commercial and open source tools.

Here are some of the tool partners our consultants have worked successfully with over the years. You may have your own favorites but it is important to ensure that any tool you use can become a useful feature of your automated DevOps system.

By utilising unparalleled automation Verifa can help you accelerate 5 key areas of the software development lifecycle:
Building. Through fast infrastructure as code, builds are optimized and parallelized through Docker instances.
Testing. Achieve test automation at unit, integration and system level. Create regression suites to verify software changes.
Static Analysis. Ensure compliance to ISO26262, FDA, MISRA, ASIL B/C.
Security. Bake in advanced security checkers, e.g. CERT, CWE, OWASP at Time Zero (T0).
Architecture. Manage software dependencies to reduce complexity, avoid erosion and optimize the build.



Key elements to consider when selecting the components of your toolchain

AUTOMATE, MANAGE, SUCCEED

There is a solid business case for moving to DevOps. At Verifa we leverage DevOps to make the most of your software deployments, including utilization of resources. We have seen projects where 70% of budgets are devoted to maintenance and operations. If you are spending too much time on maintaining software you are not focusing on being innovative and creative. And that's the raw motivation behind DevOps, to remove the latency between somebody having an idea and that idea making it through to deployment.

We have helped our clients achieve a reduction in build times by 65% and broken builds by 75% in the development branch. Our clients detect architecture defects early and reduce technical debt. One customer estimated that this saved over $100K alone. We have seen our methods reduce security vulnerabilities in deployed products by 40%.

DevOps is the way software development should always have been done. Engage with the methodology and Verifa today to strengthen your organisation for the future.

verifa

| US | Finland | Sweden | China | UK |
www.verifa.io