DevOps Transformation in an Enterprise

Contents

Executive Summary

DevOps is a great tool for **empowering your teams**. If your people are working on delivering a product or project of any kind, adopting this philosophy can help them work more effectively and **raise their morale**.

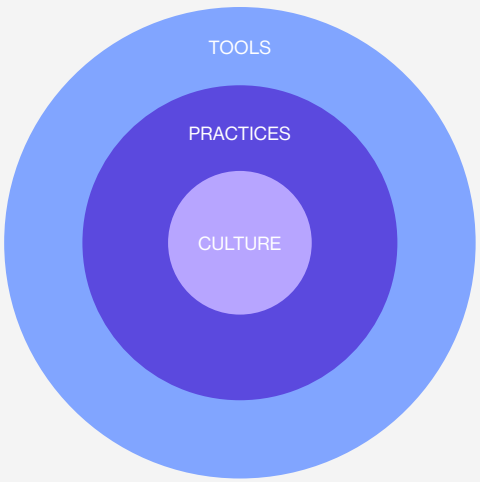
In this handbook, we present the key principles of DevOps, along with some best practices and tooling recommendations. We will also help you mitigate some of the most common challenges.

What is DevOps?

DevOps combines the best approaches, practices, and tools that bring people together to focus on meaningful work. As a result, organizations can deliver IT solutions and services more efficiently and effectively.

Think of it as a set of guidelines to follow, so your people can **focus on delivering value**, and achieving better results in the shortest time possible.

- An organization may have the most incredible toolchain in the world, but without connecting everyone to the mission and embracing experimentation, the full potential will not be truly realized.
- The core DevOps practices are: Continuous Integration, Continuous Deployment/ Delivery, Continuous Improvement, Test Driven Development, Refactoring, and Code Review.
- Services like Azure DevOps and GitHub will help put the new approach into practice, as they were built with DevOps at their core.



By implementing DevOps, organizations **break down the walls** between software development, operations, and other teams, creating a centralized, scalable environment for everyone to use in a standardized fashion.

THE RESULT?

Numerous benefits for your people, customers, and organisation - these are just the highlights:

1

People

- Happier employees
- Higher retention of top talent
- Fewer project bottlenecks

2

Customers

- Faster implementation of customer feedback
- Increased value of delivered features

3

Organization

- Reduced mean time to recovery
- Faster time to market
- Lower costs
- Deploying on demand

The four pillars of effective DevOps

1

Trust

Foster collaboration toward reaching the desired outcome by promoting interactions and welcoming input.

2

Transparency

Open communication is the foundation of strong relationships, common vision, and a shared set of goals.

3

Standardization

Use tools as accelerators to drive change based on your organization's culture and direction.

4

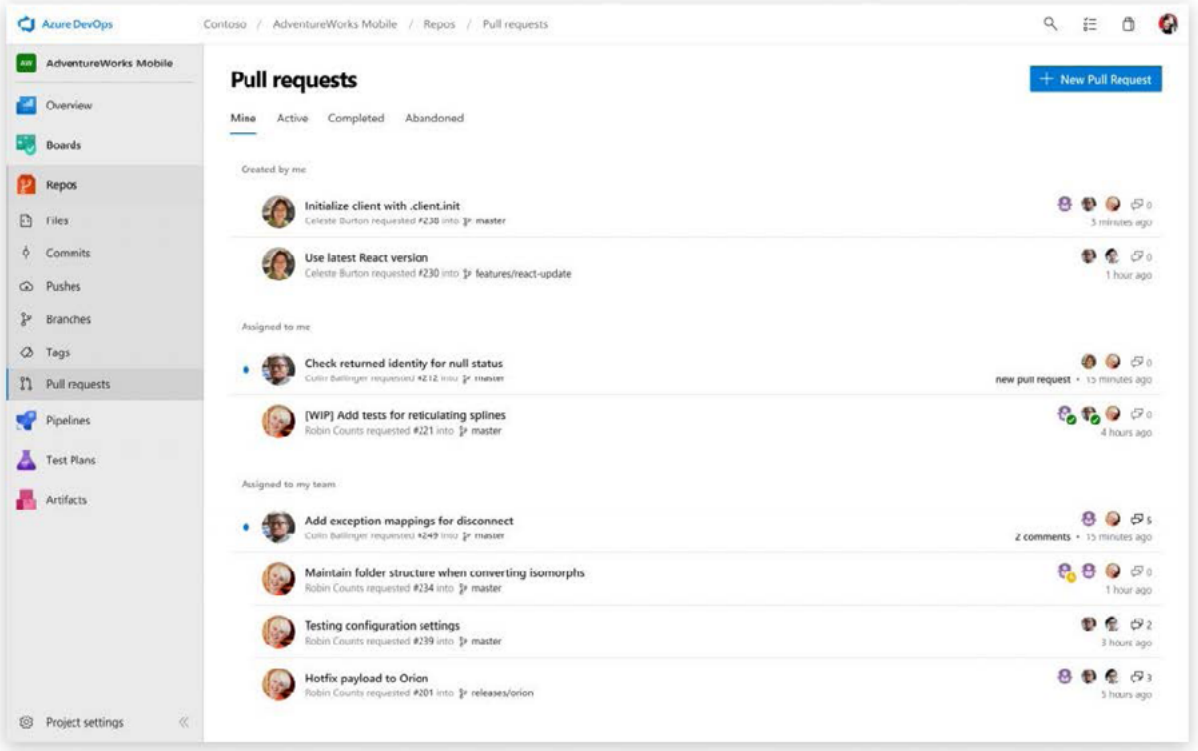
Growth

Start small and scale, gaining the ability to adopt other pillars of effective DevOps as your organization develops.

The benefits of adopting DevOps

Many organizations around the world have already invested in DevOps and are reaping the benefits as we speak. A recent State of DevOps report shows that organizations that have adopted DevOps as a practice can:

- **Increase deployment frequency** by deploying on demand
- **Reduce lead time for changes** from months to days
- **Reduce mean time to recovery** from weeks to less than 1 day
- **Improve change failure rate** from 46-60% to 0-15%



DevOps allows teams to leverage a range of verified solutions or tools to create a **consistent yet autonomous** developer experience across the organization.

This standardization leads to:

- Higher morale
- Ease of moving resources between projects
- Reduced onboarding of resources
- Centralized tooling knowledge

The five stages of DevOps evolution

Build trust among your teams through a set of shared practices.

1 Normalization

- Application development teams use version control
- Teams deploy on a standard set of operating systems

2 Standardization

- Teams deploy on a single standard operating system
- Build on a standard set of technology
- Security teams are involved in technology design and development*

3 Expansion

- Individuals can work without manual approval outside the team
- Deployment patterns for building apps/services are reused
- Infrastructure changes are tested before deploying to production*

4 Automated infrastructure delivery

- System configurations are automated
- Provisioning is automated
- System configurations are in version control*
- Infrastructure teams use version control*
- Application configurations are in version control*
- Security policy configurations are automated

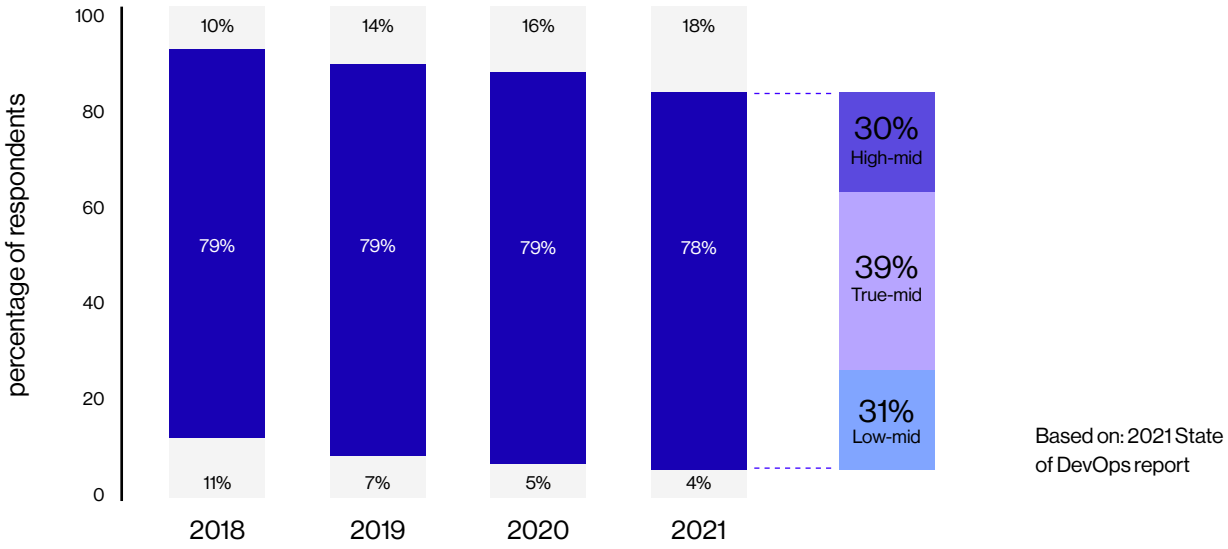
5 Self-service

- Incident responses are automated
- Resources are available via self-service
- Applications are rearchitected based on business needs*

* these practices are highly correlated with the stage. Based on: 2019 State of DevOps report

DevOps adoption in organizations

Most companies introduced the approach to some extent but got stuck in their journey.



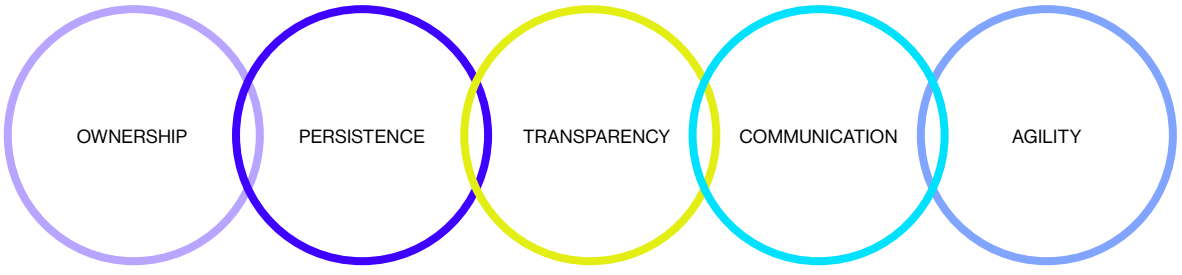
Based on: 2021 State of DevOps report

CULTURE

Cultural shift is fundamental

Collaboration is one of the most important pillars of DevOps. Team interaction and collective input are critical in working toward a goal.

DevOps culture promotes qualities such as:



It's no secret that employees who are both motivated and empowered **do great work**

Processes help structure and evaluate your efforts

Standardization can sometimes hinder flexibility but finding the right balance dependent on your situation may help your teams work more autonomously.

Here are some practical hints for helping your teams work better together:

- Use version control and trunk-based development
- Involve stakeholders early in the development process
- Establish reusable deployment patterns (e.g., automated pipelines, Infrastructure as Code)
- Automate testing, compliance activities, security scanning, and approvals
- Share source code with other teams, using a centralized library repository
- Continuously gather feedback from customers and stakeholders, and make improvements
- Know who your customer is - they're not always external to your organization

TOOLS

Whatever you do, no matter the industry, it is best to have the right tool for the job.

Using specialized services like GitHub or Azure DevOps, you can:

- Support teams in planning their work,
- Collaborate on code development,
- Build and deploy applications quickly and efficiently,
- Automate reporting, so developers can focus on meaningful work and creating value.

Working with DevOps tools

Microsoft Azure DevOps and GitHub, whether used independently or as part of a larger integrated solution, provide developers with services that help teams plan, track, collaborate and deploy applications.

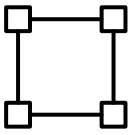
Azure DevOps

Let's start with the key features your teams can use for their projects.

Planning and tracking

Microsoft Azure Boards is known for strong analytics, dashboards and flexible management frameworks to support everything from nested backlogs and cross-team cadence tracking to enterprise level reporting.

The power of Azure Boards lies in its strong analytics engine, dashboards, and complete customization of the Process and Work Items to fit the needs of your organization.



Workflow automation

Azure Pipelines gives organizations the ability to create automated pipelines for CI/CD at scale, including advanced test reporting.

For this reason, Azure Pipelines continues to be the best fit for organizations with complex deployments. It supports multi-stage pipelines with fine-grain permissions, templates, approvals, gates, and custom checks to enforce security, compliance, and safe deployment best practices.

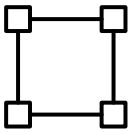
GitHub

Now let's see how these functionalities are covered by GitHub.

Planning and tracking

GitHub Issues are ideal for small team, community-supported projects that require loose tracking and flexible bug management.

With the addition of Projects, GitHub also became more scalable and can be used comfortably within enterprises. You can now take advantage of features like manual approvals in workflows, workflow visualization, and policy enforcement.



Workflow automation

GitHub Actions offers a level of flexibility, ease of automation and a deep, rich community library that will create an immediate increase in efficiency.

It allows teams to create automated workflows and build, test, and deploy applications and services.

While many teams will choose to create their own, some choose to use one of the thousands of tested actions from the community and trusted partners like HashiCorp, Ansible, CodeCov, Launch Darkly, and many others available on GitHub Marketplace.

Notable differences

Let's take a look at some points where Azure DevOps and GitHub are different from each other.

	GitHub	Azure DevOps
Version Control	GitHub is a distributed version control system. It provides seamless workflows that connect open source with your DevOps processes. You can use it in Visual Studio, your favorite coding tools, or through Codespaces in your browser. It is also equipped with robust, constantly developed security features. Tools like CodeQL, Dependabot, and Native Scanning allow you to shift your security left and protect your code at the source.	Azure Repos is a set of tools that makes version control possible by enabling developers to collaborate and review code, and to manage source code. It also stores the entire change history of your code base, so it is always easy to get a specific version when needed. Azure DevOps offers two version control systems: • GIT • TFVS (Team Foundation Version Control) Your people can access Azure Repos through different Integrated Development Environments (IDE), including Visual Studio or XCode.
Testing	Microsoft is currently working on bringing testing functionality to GitHub.	This is an area where Azure DevOps really shines. It's very simple to build out fully integrated solutions between exploratory, manual, and user acceptance testing, and Test Plans that tie in directly to the workflow of product and project management. Best of all, Azure Test Plans can be fully leveraged from the browser which reduces cost in IDEs and onboarding personnel.

	GitHub	Azure DevOps
Packages	GitHub Packages include a great variety of feeds to choose from. Auditing features are not yet available but basic functionalities are included in Container Registry. Granular anonymous access to managed packages isn't available, but you can make packages private or public. Public packages can be accessed anonymously. GitHub Packages and Container Registry are not available for private repositories in accounts with legacy per-repository plans.	Azure DevOps includes Azure Artifacts, allowing you to use many different types of package feeds from both public and private sources. These include Python, NuGet, Maven, npm, etc. Azure Artifacts is the likely choice if you have strict auditing requirements, require granular security, or anonymous access.
Insights and Analytics	GitHub Insights focuses on measurement, analytics, and transparency during every stage of software development. It allows greater focus on developer productivity, optimization of software delivery, and understanding the depth and effectiveness of inner source collaboration. GitHub Insights is available for GitHub Enterprise Server (GHES) with support for repo and organization insights. GitHub is continuously expanding the functionality and capability of Insights to include measurements of the most impactful industry benchmarks.	Azure DevOps provides many ways to take a deeper look into software development, managing Portfolios, testing, and CI/CD workflow. You can use Azure Analytics, Dashboards, Widgets, ODATA feeds, or exported SQL data views which can be consumed by 3rd party tools. When adding Power BI on top of this strong set of tools you can measure success in just about any way you can imagine.

Integrating both tools

Using the integration tools released by Microsoft, you can link work items with commits in GitHub directly to Azure Boards and track those changes within the state of the work items themselves.

Note: Azure Pipelines cannot track the work items through all stages of CI/CD. If you need full visibility into every part of your CI/CD pipeline, use one of the two following combinations:

- Azure Boards, Azure Repos, and Azure Pipelines or
- GitHub Issues, GitHub Repos and Azure Pipelines

It's important to remember that GitHub offers additional functionality that doesn't get enough press. It's the ability to have a running and dynamic conversation between the team and the community using Discussions.

Discussions is a powerful mechanism for incorporating your customer feedback and bug reporting back into product planning and development. You can leverage API solutions to bring the Discussions back into your Azure Boards work items and incorporate that feedback into your planning. Using this method, you have the ability to automate part of your user interview or customer feedback process directly into Azure Boards.

Which tool should I choose?

The recommendation will largely depend on what you currently use at your organization, what your people are familiar with, and the scale of your projects. Still, there are some broad suggestions, based on the roadmap of both tools.

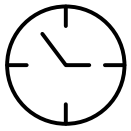
If you

- work on GitHub
  you're already set!
  Keep discovering new functionalities to maximize your investment.
- work on Azure DevOps
  don't worry, it's not going anywhere anytime soon.
  Keep your backlog and task management in Azure DevOps, so you retain your project history. At the same time, start thinking of introducing GitHub for new or small projects.
- don't use either tool
  start with GitHub
  Your teams will work in a familiar environment and you will take advantage of the latest features and improvements.

KEY DEVOPS ADOPTION CHALLENGES

What do organizations struggle with the most?

Here are the top 5 challenges that stand in the way of adopting, along with recommendations for overcoming them.



Organizational Change

DevOps transformation, like any change, can bring many associated challenges, such as morale decrease, fear, or higher need for support.

How to mitigate it?

- Take care of your teams by showing empathy and care
- Fail fast and learn from the experience
- Reward the delivery of value, collaboration, and growth

Lack of inclusion of non-DevOps teams

Difficulties in communication and no clear sense of ownership can lead to slower approvals and disrupt project flow.

How to mitigate it?

- Prepare, plan, and support your teams to show trust and reduce wait times
- Focus on Continuous Improvement and automate approval processes
- Connect cross-organizational stakeholders early

Disconnect between executives and practitioners

Without a clear direction and awareness of the risks, even the best attempts can fail. You need to understand the impact of DevOps to measure its value appropriately.

How to mitigate it?

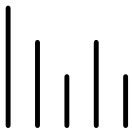
- Prepare, plan, and support your teams to achieve a common goal
- Start small and scale
- Focus on capabilities, not maturity

Customer focus

It's easy to get lost in the process and forget about the main purpose, which is delivering value for the end-user.

How to mitigate it?

- Incorporate feedback gathering into the development process from the start
- Build feature measurement into the product from the beginning

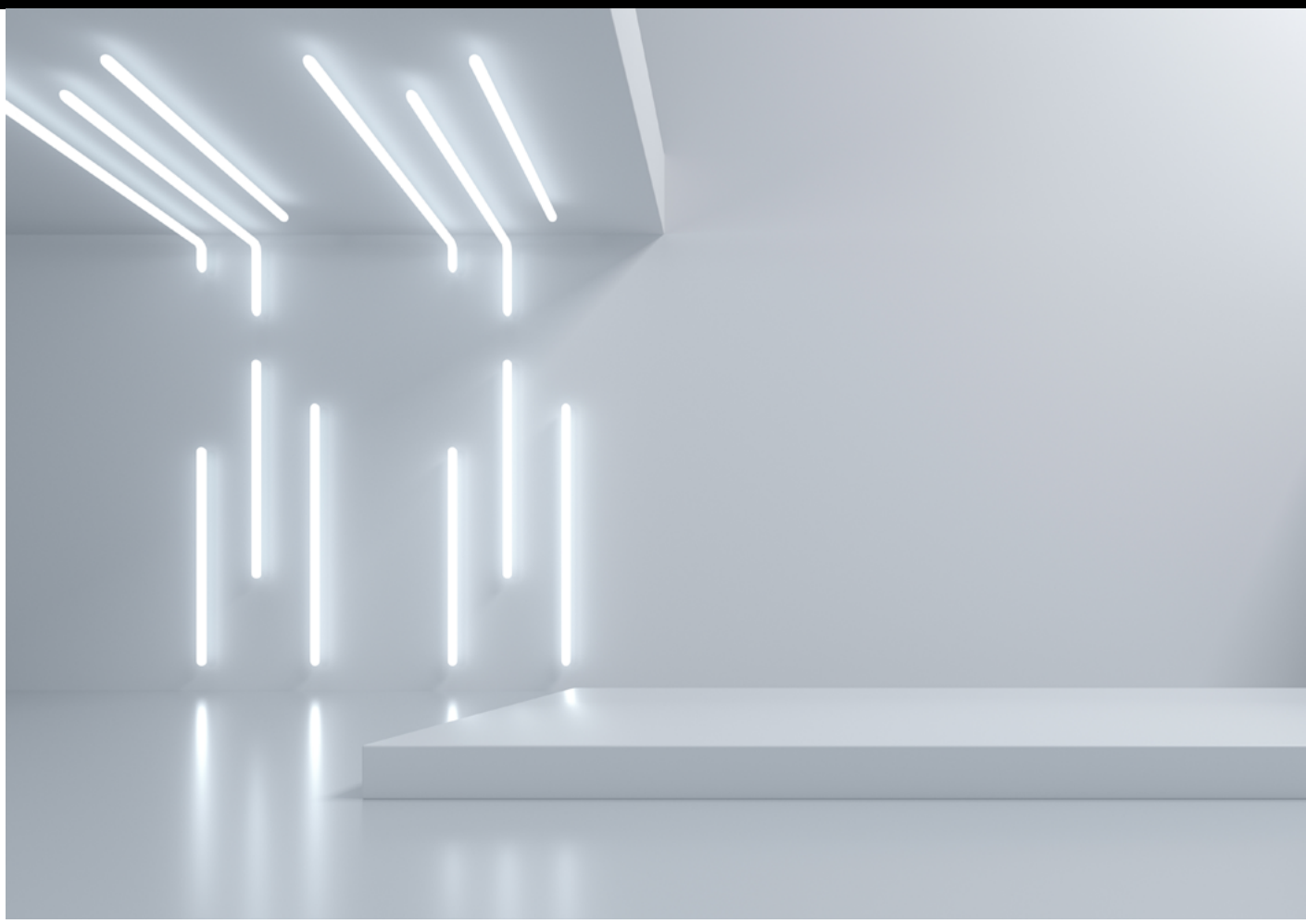


Inconsistent standardization

Any tech stack needs consolidation of licensing and costs, so you can minimize expense, reduce technical debt, and retain the best tools for the job.

How to mitigate it?

- Build an Enterprise DevOps Framework (EDF) to support all stakeholders and optimize costs
- Provide guardrails for your teams without dictating full toolchain requirements
- Incorporate stakeholder feedback from all departments



SUMMARY

DevOps is not one tool or one team.

It is not a single practice or process. DevOps is a combination of cultural philosophies, practices and tools that increases an organization's ability to quickly and efficiently deliver IT solutions and services.

Azure DevOps and GitHub are both great services with a wide range of capabilities to help you maintain DevOps practices. Because they are easily integrated, it's likely you could find yourself using a combination of tools and functionalities from both platforms.

CONTACT US TODAY

Find out more at
www.SoftwareOne.com

SoftwareOne AG | Headquarters
T. +41 44 832 41 69
E. info@SoftwareOne.com