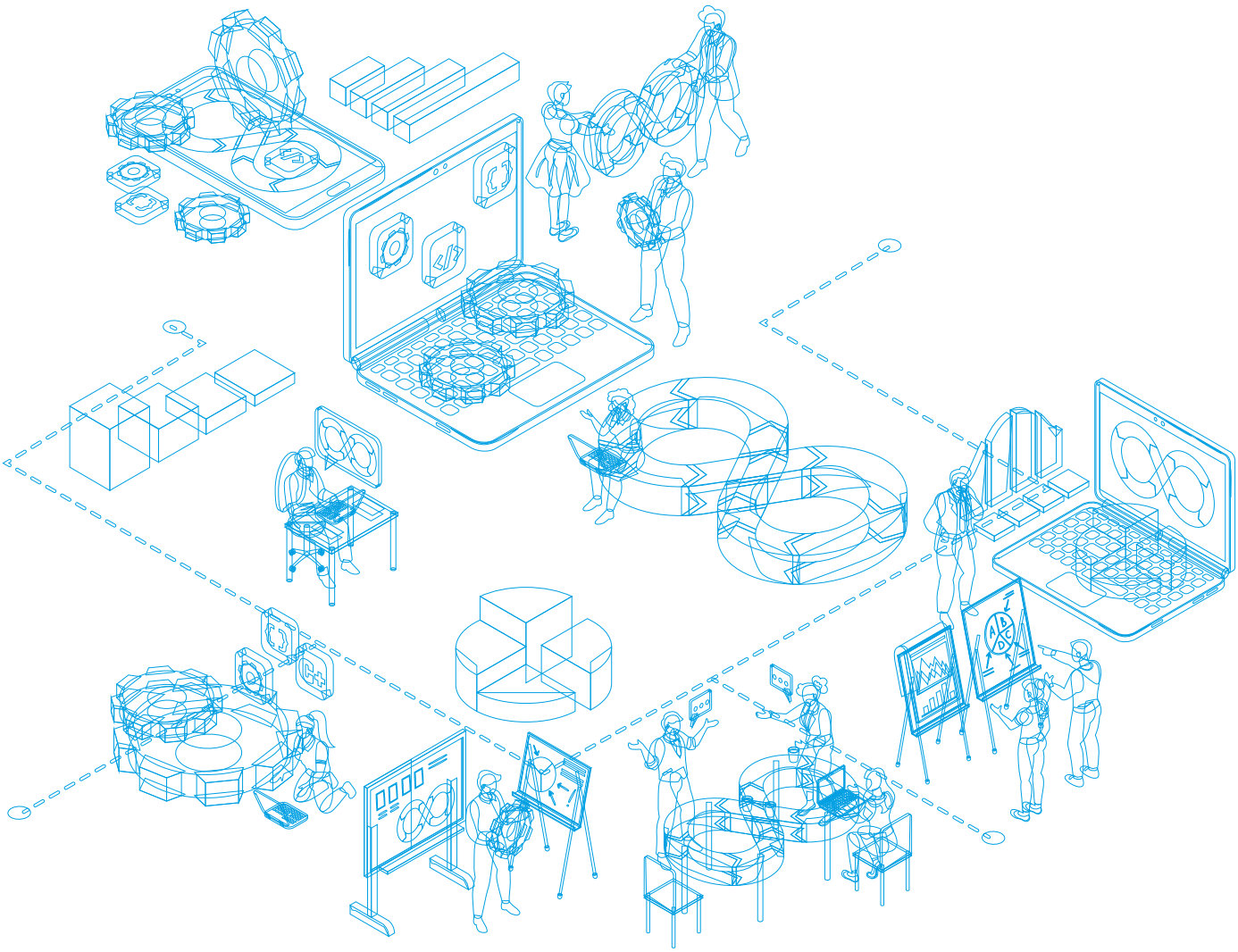
CODIFICATION

What is Cloud Native DevOps Maturity?



July 2022

Field Guide

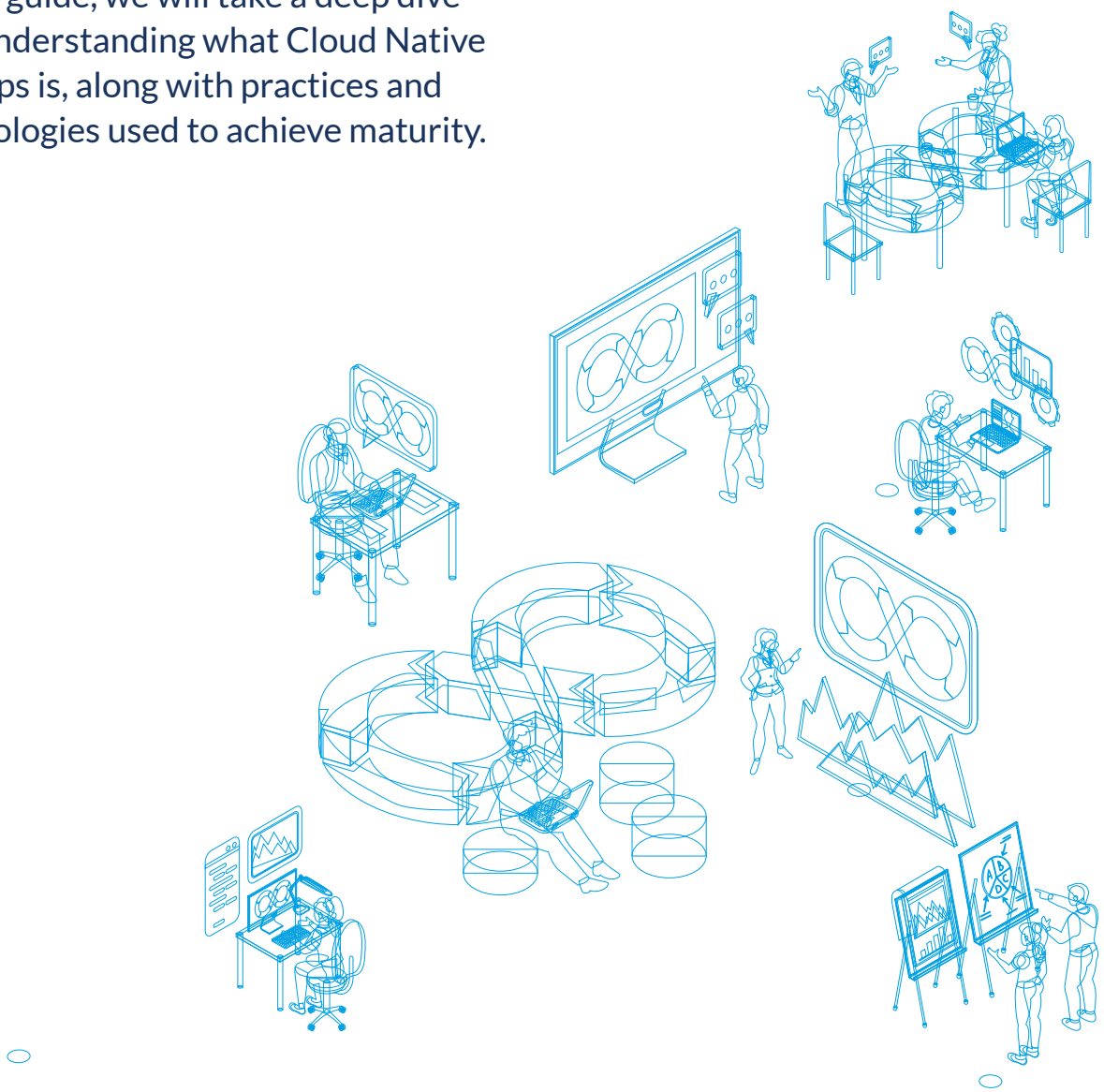
Table of Contents

Introduction

Digital transformation is a strategic goal for many organisations today. As a regulated enterprise, it's even more critical to prevent your business from being disrupted by new startups armed with better technologies and more data. The way to achieve complete modernisation through digital transformation is by adopting cloud native technologies and implementing DevOps processes in your organisation. In technical terms, we call this target state of transformation as Cloud Native DevOps Maturity.

Cloud Native DevOps Maturity essentially covers transforming the technologies, processes, policies, and people in your organisation to achieve full modernisation.

In this guide, we will take a deep dive into understanding what Cloud Native DevOps is, along with practices and technologies used to achieve maturity.



What is Cloud Native?

Cloud native is essentially a modern way of developing software. It uses many modern techniques in software development such as microservices, containers, CI/CD pipelines, agile methodologies, DevOps and DevSecOps practices.

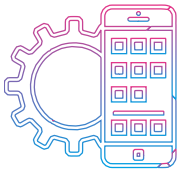
Adopting cloud native gives organisations the ability to harness the flexibility, scalability, and robustness of cloud computing. The practice of cloud native uses many tools and techniques to build applications and platforms for public cloud services such as Google Cloud, Microsoft Azure, or Amazon Web Services - as opposed to using traditional methods such as on-premises data centres.

The cloud native approach has essentially been powered by new-age businesses offering more customer-centric, user experience-focused services such as Uber, Netflix, Spotify, and Airbnb. Today, the cloud native approach is being adopted by companies that are looking for better digital agility and disruptive capabilities, essentially giving them a significant competitive advantage over their competitors.

The Cloud Native Computing Foundation (CNCF) describes cloud native as below:

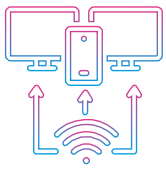
"Cloud native technologies empower organisations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach."

Understanding the Landscape of Cloud Native



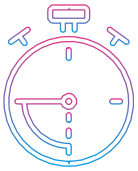
Application Definition and Development

This is the topmost layer. This stage focuses on tools used by developers for cloud native projects. These tools can be used to build new cloud native applications (databases, VOIP systems, containers, etc.) and platforms. They are also used in building CI/CD pipelines.



Provisioning

This layer focuses on everything that is needed to build and securely deploy the environment in which a cloud native application would run. Developers implement techniques such as infrastructure-as-code, imaging, build automation, ensuring application security and key and policy management in this stage.



Runtime

This is the layer on which everything related to running a cloud application is focused. Developers concentrate on containerisation with tools such as Docker, storage and networking in this stage.

Orchestration and Management

This is the layer on which everything related to running a cloud application is focused. Developers concentrate on containerisation with tools such as Docker, storage and networking in this stage.

In addition to the above layers, other practices and techniques such as Observability need to be considered as well. As the complexity of a cloud native environment increases, organisations can purchase additional resources from the market such as platform-as-a-services (PaaS) services.

Why Cloud Native?

Cloud native approach has many advantages over the traditional on-premises approach.

Cloud Native

Affordability - Cloud has recurring fees instead of an upfront fee. Although the monthly cost could be higher, maintenance and support services are already included, requiring no annual contracts.

Predictability of Cost - Gain from monthly payments that are predictable and cover software licenses, upgrades, support, and daily backups.

Less Maintenance - You don't have to worry about server maintenance as cloud software is hosted on public servers.

Better Security - Your data is safer on the cloud than in an on-premise server because data centres employ security procedures that are more expensive for most firms to pay.

Quicker Deployment - Unlike on-premise programs, which must be installed on a physical server and each PC or laptop, cloud-based software is distributed over the Internet in a matter of hours or days.

Less Energy Use - Cloud technologies offer more flexibility as you only pay for what you use and can simply scale to meet demand, for example, by adding and removing licenses.

Better Scalability - Greater flexibility is offered by cloud technologies because you only pay for what you use and can simply scale to meet demand, for example, by adding and removing licenses.

On Premises

Total Cost of Ownership - The Total Cost of Ownership (TCO) of an on-premise solution is lower than that of a cloud system because you only have to pay for your user licenses once.

More Control - You own all of your data, hardware, and software platforms. You make the configuration, upgrades, and system modifications.

Less Dependability - With on-premise systems, you may access your software without relying on the internet or other outside variables.

However, they both have their own disadvantages.

Cloud Native

Need for Connectivity - To use cloud solutions effectively, you must have dependable internet access.

Higher Costs Long-Term - Cloud applications can be more expensive over the duration of the system's life cycle despite needing a smaller initial investment, raising the total cost of ownership (TCO).

Less Flexibility - You don't have to worry about server maintenance as cloud software is hosted on public servers.

On Premises

High-Cost - Since on-premise systems typically require a sizable initial investment, capital expenditures are frequently necessary. In addition, you must factor in maintenance expenses to guarantee support and functional improvements.

High Maintenance - With an on-premises system, you are in charge of disaster recovery, data backups, storage, and server hardware and software. Smaller businesses that have constrained financial and technological resources may find this to be a problem.

Implementation Takes Time - Because each computer and laptop must have its own installation process, on-premise implementations take longer.

Due to its adaptability, dependability, and security, cloud native can be regarded as being better than on-premise since it frees you from the headache of maintaining and updating systems and lets you focus your time, money, and resources on executing your essential business strategy. With increasingly more adoption rates, cloud services are quickly overtaking other options for organisations since they offer real-time access to systems and data from a range of devices regardless of

What is DevOps?

How Does DevOps Work?

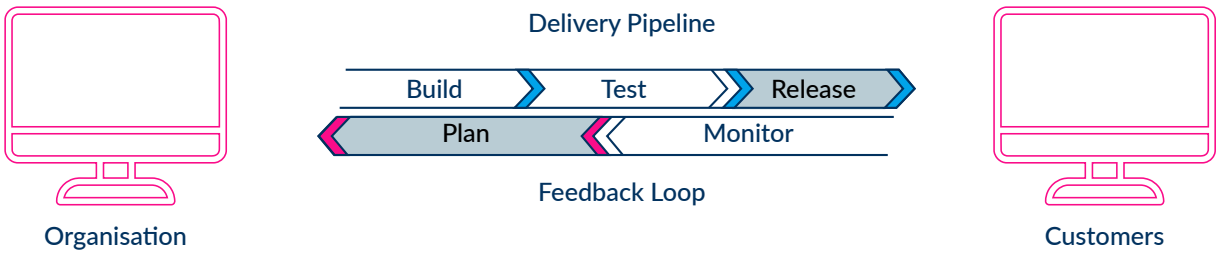
In traditional ways of working, teams in an organisation work in a “siloed” manner; which means teams or departments operate on their set prerogatives with little collaboration with each other.

However, with DevOps, this is not the case. Teams in a DevOps culture will sync with each other on a real-time basis. Additionally, teams often get merged where engineers for instance will work across the entire lifecycle.

DevOps models often integrate multiple functions into single entities to increase efficiency and cooperation. For example, in some models, quality assurance and security teams can work together alongside development and operations teams across the software development lifecycle.

DevOps teams also use tools to automate repetitive tasks that traditionally were manual and slow. They also use new technologies that make it easier to update and evolve applications more quickly. With new tools, technologies, and ways of working, DevOps enables developers to work more independently and do tasks with more velocity without requiring help from other teams.

Defining the DevOps model



Benefits of Adopting DevOps

Speed:
Increasing the velocity of the software development process enables you to innovate and deliver products for your customers faster. It also makes it possible to adapt to changing markets better.

Reliability:
Using practices like Continuous Integration and Continuous Delivery means you get to increase the reliability of your pipeline while also improving its quality. You can also assess the performance of your applications in real-time by using practices such as Monitoring and Logging.

Better corporation:
DevOps promotes a working culture based on accountability and ownership. In a DevOps environment, developers and operations teams work collaboratively while sharing responsibilities that help reduce inefficiencies and save time.

Better delivery times:
DevOps makes it possible to increase the frequency and velocity of new product releases. The faster you can iterate and release new features and fix bugs, the better you will be able to be more competitive by responding to customer needs effectively.

Scale:
DevOps brings consistency to the table. By combining consistency with practices like automation, managing dynamic and complex systems becomes easier with low risk. For example, using a technique like infrastructure-as-code enables you to increase the repeatability and efficiency of your software development lifecycle substantially.

Improved security:
In a DevOps model, you can use practices such as automated compliance policies, fine-grained controls, and configuration management techniques to significantly improve the security of your applications.

Why DevOps is so Crucial Today

The world has been completely transformed by software and the Internet. Software now plays a crucial role in every aspect of a business, going beyond simply providing support. Through software supplied as online services and applications on a variety of devices, businesses can scale their communication and have more visibility into their customers than ever before. Additionally, they use software to transform every step of the value chain, including logistics, communications, and operations. Companies now must change how they produce and distribute software in a similar way to how physical goods companies changed how they design, build, and deliver things utilising industrial automation throughout the 20th century.

Understanding DevOps culture

Changes in an organisation's culture and mindset are necessary to make the transition to DevOps. DevOps is essentially about breaking down barriers between development and operations - where traditionally, the two teams operated in silos.

DevOps increases the efficiency of developers and the dependability of operations. The teams make an effort to communicate regularly, boost productivity, and raise the calibre of the services they offer to consumers.

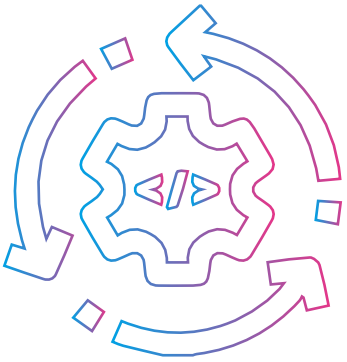
Teams for quality control and security may also become closely incorporated with these teams.

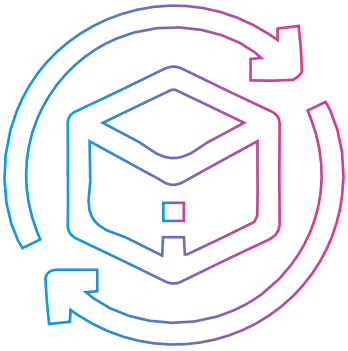
Regardless of their internal structure, organisations implementing a DevOps model have teams that see the full development and infrastructure lifecycle as a part of their duties.

DevOps Practices

Continuous Integration

Software engineers who use Continuous Integration constantly merge their code changes into a common repository, which is followed by automated builds and testing. Continuous Integration's main objectives are to detect and fix issues more quickly, enhance the quality of software, and shorten the time it takes to validate and publish new software upgrades.



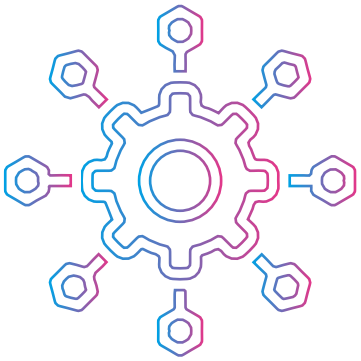


Continuous Delivery

Continuous Delivery involves automatically building, testing, and getting code updates ready for production release. By delivering all code alterations to testing environments and/or production environments after the build step, it advances on Continuous Integration. When Continuous Delivery is implemented correctly, developers will always have a build artefact that is ready for deployment and has undergone a set of tests.

Microservices

A single application can be built as a collection of small services using the microservices architectural design strategy. Each service operates as a separate process and interacts with other services via a clear interface and a lightweight method, usually an HTTP-based API. Microservices are based on business capabilities, and each service has a narrow scope. Microservices can be created using a variety of frameworks or programming languages and deployed individually, as a single service, or as a collection of services.



Infrastructure-as-Code

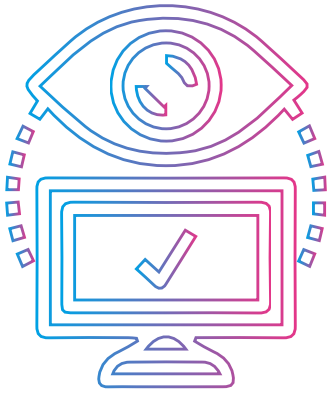
In the practice of "infrastructure as code," infrastructure is created and controlled through the use of code and software development methods like version control and continuous integration. Instead of needing to manually set up and configure resources, developers and system administrators may interact with infrastructure programmatically and at scale thanks to the cloud's API-driven paradigm. Because of this, engineers may interact with infrastructure using tools that are based on code and treat it similarly to how they approach application code. Infrastructure and servers may be swiftly deployed using standardised patterns, updated with the most recent fixes and versions, or duplicated in repeatable ways because they are defined by code.

Configuration Management

Code is used by programmers and system administrators to automate operational operations, host and operating system settings, and more. Configuration changes are repeatable and standardised when code is used. Developers and systems administrators are freed from having to manually configure servers, operating systems, and system applications.

Policy-as-Code

Organisations may monitor and enforce compliance dynamically and at scale when infrastructure and its configuration are formalised in the cloud. Code-described infrastructure may therefore be tracked, verified, and modified automatically. This makes it simpler for enterprises to control resource changes and guarantee that security policies are correctly applied across the board (e.g. information security or compliance with PCI-DSS or HIPAA). Since non-compliant resources can be immediately marked for more examination or even brought back into conformity, this enables teams inside an organisation to move more quickly.



Monitoring and Logging

Organisations keep track of metrics and logs to see how infrastructure and application performance affects how customers use their products. Organisations can gain insights into the underlying causes of issues or unanticipated changes by capturing, classifying, and then analysing the data and logs generated by applications and infrastructure. As services must be accessible around-the-clock and as the frequency of application and infrastructure updates rises, active monitoring becomes more and more crucial. Organisations can monitor their services more proactively by setting up alerts or doing real-time analysis of this data.

Communication and Collaboration

One of the most important cultural aspects of DevOps is the improvement of communication and collaboration inside an organisation. By physically combining the tasks and workflows of development and operations, collaboration is established through the use of DevOps technologies and automation of the software delivery process. On top of that, by utilising chat applications, issue or project tracking systems, and wikis to facilitate communication, these teams established strong cultural norms on information sharing. As a result, communication between developers, operations, and even other teams like marketing or sales can be sped up, enabling a closer alignment of objectives across the entire organisation.

What Cloud Native DevOps Maturity Looks Like

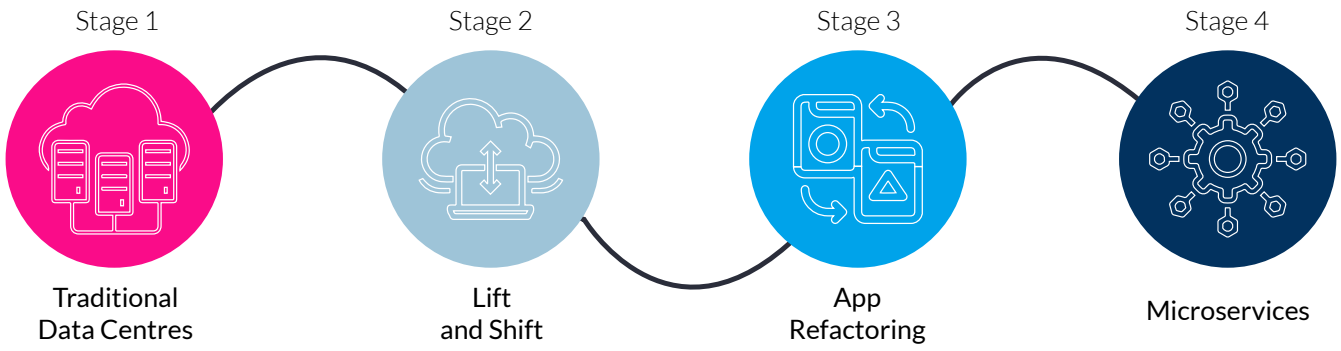
In previous sections, we explored what cloud native is, what DevOps is and why Cloud Native DevOps is so critical to modern digital transformation. When an organisation is fully modernised with Cloud Native DevOps, we say that that organisation has achieved Cloud Native DevOps Maturity.

Let's explore how the process works and what the result of adopting Cloud Native DevOps maturity looks like

Cloud Native Maturity

An organised and thorough migration model along with a wise investment plan will go a long way toward guaranteeing success. An effective approach for achieving cloud native maturity will pinpoint typical adoption patterns or "maturity levels." You may find it simpler to create your own cloud approach if you comprehend these steps.

For this guide, let us now discuss four general levels of achieving cloud native maturity.



It's crucial to remember that advancement doesn't necessitate adopting every step along the route. Some businesses skip stages entirely, moving straight from Stage 1 to Stage 3, for example. Not every firm needs to reach Stage 4, and the majority will simultaneously have a variety of apps in their portfolio at various maturity levels. For instance, it makes sense to shift customer-facing apps that generate money to Stage 3 or Stage 4, which are geared for quick application delivery and scaling. These levels need to be the most agile and responsive. Older programs that are in maintenance mode, on the other hand, can be maintained at Stage 1 or Stage 2 without needing to make a significant further investment. Remember that businesses are implementing a DevOps operational philosophy to complete these transformational objectives as well. Hybrid apps are also a reality, where some components of the program remain to function on-premises because of fixed mainframe systems or data gravity.

Stage 1: Traditional Data Centres

Traditional data centre apps are often watched over by the IT Ops team and run in traditional virtual machines (like VMware) or on bare metal in a private data centre. Of course, there are advantages and disadvantages to various deployments and structures.

Total control over your hardware, software, and data, reduced reliance on outside elements like internet access, and perhaps a lower total cost of ownership (TCO) when applied at scale are all benefits.

The drawbacks include high initial expenses that frequently necessitate capital expenditure (CapEx), as well as maintenance obligations.

Stage 2: Lift and Shift

A cloud adoption journey has been started by many organisations today due to the elasticity and agility that the cloud offers. Cloud migration, however, rarely goes without a hitch. A company will frequently “lift and move” the identical virtual machines it was using on-premises to the cloud. Usually, the target environment is a public cloud service provider like Amazon AWS, Microsoft Azure, or Google Cloud, but it can also occasionally be a private data centre set up as a private cloud.

Despite the anticipated cost advantages of a strong migration strategy, businesses frequently discover that running their apps in the cloud, in the same manner, it did on-premises is significantly more expensive.

These programs were designed for huge VM configurations, which are quite expensive on the cloud. In other instances, the infrastructure's support is absent from the premises. For instance, when simply lifted and relocated to the cloud, several application servers that depend on multicasting capabilities to connect no longer function. A refactoring of the application is necessary due to these flaws.

Finally, these traditional data centre applications were developed based on hardware dependability hypotheses that might no longer hold in the cloud. In contrast to the cloud, which is established on the concept that hardware is cheap and unreliable and that the software layer provides application durability and stability, on-premises hardware is often custom manufactured and intended for superior reliability. This is yet another element that necessitates application refactoring.

Stage 3: App Refactoring

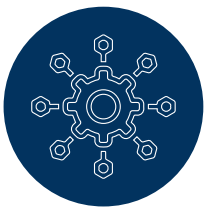
An enterprise must modify its apps once it finds that certain of its lifted-and-shifted applications are fundamentally constrained in the cloud. Refactored programs are a fantastic fit for many contemporary platforms that are made specifically for executing apps in cloud environments.

These contemporary platforms frequently include cloud-native services or application platform-as-a-service (PaaS) technology. Pivotal Cloud Foundry, Red Hat OpenShift, AWS Elastic Beanstalk, Azure PaaS, and Google App Engine are examples of common PaaS technologies. Numerous managed services from AWS, Azure, and Google Cloud are included in the cloud native offerings, including messaging, Kubernetes, and database services, to mention a few.

The deployment, management, and scaling of the application code are under the purview of the IT group in a traditional data centre apps scenario. These contemporary platforms, in contrast, automate processes and abstract away a lot of complexity. Applications become elastic, dynamically utilising computing resources to accommodate changing workloads. The current method frees the IT department from managing items that offer no intellectual property benefits to the company, allowing it to concentrate on business challenges rather than infrastructure difficulties.

Programs that have been refactored frequently contain the same code as monolithic apps but in smaller, easier-to-manage pieces. Although the code for these smaller components was not initially designed for completely modular, stateless deployments that would make them linearly scalable, they can now scale automatically using PaaS or cloud-native services.

Stage 4: Microservices



As the name implies, a microservices architecture divides an application into several small, modular, and independently deployable services that may scale as needed to handle the demands of an application's workload. These services are typically designed utilizing serverless computing or container and orchestration technologies (such as Docker and Kubernetes or AWS, Azure, and Google container services) (AWS Lambda, Azure Functions, or Google Functions).

Applications created using microservices architectures are extremely scalable and fault resistant, guaranteeing a flawless end-user experience. Smaller services can be given to agile DevOps teams by organisations, allowing them to operate freely and develop more quickly, bringing new products to market faster.

However, it necessitates a significant investment: either the organisation must redesign existing applications from the ground up or use the microservices method when creating any new apps.

Even while these granular services have numerous advantages and are easier to administer individually, merging them into larger business applications can be challenging, especially in terms of monitoring and troubleshooting.

Cloud Native Also Needs DevOps to Work

The success of the adoption journey depends not only on technology but also heavily on organisational transformation. To reap the full rewards of reaching greater levels of maturity, siloed Dev and IT organisations must be replaced with more DevOps-oriented ones.

A DevOps team usually includes developers and engineers for site reliability (SREs). The entire application is not the responsibility of each agile team; rather, it is just a few services. Teams take tremendous satisfaction in how quickly they can offer new functionality and how responsive their services are. Extreme ownership is the mantra here. DevOps differs from the conventional Dev and Ops divide because of this.

DevOps Maturity

Instead of viewing DevOps as a continuous journey of development and operations optimisation and integration, many organisations make the error of treating it like a final, static destination. However, with a structured approach, you can visualise a roadmap for achieving DevOps maturity for your organisation.

With such an approach, you can more easily advance to the next level by determining the strengths and weaknesses of your organisation and concentrating your training on the critical areas that require development.

Fundamentals of DevOps

1. Culture

Technology is only one aspect of DevOps; a change in company culture is also necessary. The support of all stakeholders, from engineers to executives, is necessary for your DevOps journey. Effective cross-functional partnerships are necessary for DevOps maturity, as are business leaders' consistent and reasonable expectations.

2. Testing

The distinction between development and testing does not exist in a mature DevOps ecosystem. Every product should have its own testing environment, and testing should be automated. Additionally, to make sure you're not leaving any gaps, you should frequently do risk evaluations as well as ongoing analysis and validation of your test coverage.

3. Automation

To achieve the continuous delivery and deployment schedule necessary for DevOps maturity, automation is essential. This is commonly known as CI/CD, or continuous integration/continuous deployment, in DevOps. Continuous integration and continuous delivery (CI/CD) aims to improve software quality without slowing down development by anticipating and eliminating problems through continuous testing throughout the software development lifecycle.

4. Architecure

You require application architecture that is created to meet your DevOps objectives to move up the DevOps maturity model. With distinct modules that can operate (or fail) without affecting the work of others, explicitly stated quality criteria, and protection against cascade failures, your design should facilitate simple testing and quick deployments. You must select an architecture that meets your needs and is consistent with your DevOps maturity goals because no single design is suitable for all DevOps settings and infrastructure.

Stages of Achieving DevOps Maturity

On the path to DevOps maturity, an organisation will typically pass through five stages. Although you can't skip any levels, you'll probably discover that each one is exponentially easier to reach as you start to worry less and less about details and concentrate more on higher-order processes.

Stage 01: Status Quo

Stage 1 places you in a conventional IT environment where operations and development are managed separately. Even if you could be considering a switch to DevOps, you haven't yet put any changes into practice.

Stage 02: Implementation

When your operations teams start focusing on automation and your development teams start putting more of an emphasis on obtaining higher agility, you have entered stage 2. You're beginning to stress collaboration between the development and operations teams in stage 2.

Stage 03: Development

In stage 3, organisations install well-defined automation and DevOps procedures into place. Additionally, your entire organisation must join the DevOps journey, accept these new procedures, and support your objectives.

Stage 04: Iteration

Your company is fully committed to DevOps methods and understands them. Your current priorities should be increasing your CI/CD and optimizing your DevOps.

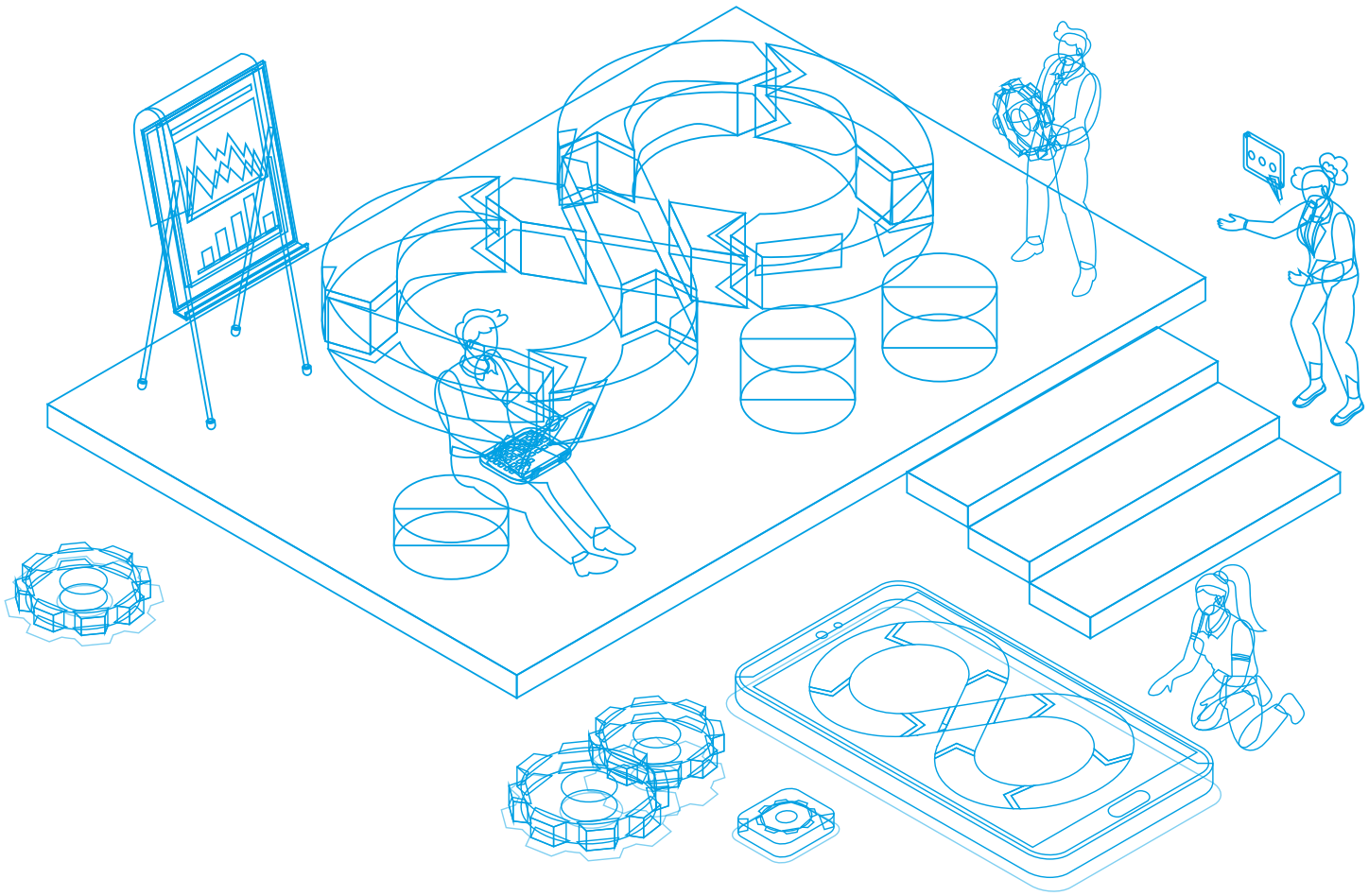
Stage 05: Target State

Your DevOps procedures are now producing observable outcomes. Even though you continue working to enhance collaborative teamwork and process optimisation, you're still able to celebrate successes at the corporate, team, and individual levels.

What's next for your DevOps Roadmap?

Your DevOps journey will never be over, and to maintain DevOps maturity, you'll need to continually look for ways to make your procedures more efficient. As you move through the levels and improve your processes, moving through the DevOps maturity model becomes easier. However, to avoid stagnation, you still need to continually assess your abilities and development.

You might need to hire outside DevOps professionals to help you better utilise the tools and processes necessary to get you back on track if you find yourself stuck at a certain level and unable to pinpoint the areas you need to improve to advance.



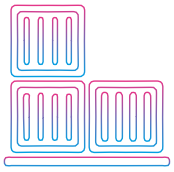
How Do You Get Started With Cloud Native DevOps?

The starting points for a cloud-native strategy are as follows:

1. Selecting a cloud provider over an on-premises one to implement a cloud-first strategy
2. Adopting a multi-cloud strategy
3. Instilling a DevOps culture

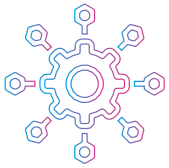
DevOps becomes a priority when businesses realise that agile development requires both automation and a change in culture to build quality apps more quickly. It might become quite difficult to manage many hybrid setups or streamline infrastructure-related tasks. Because of this, the use of technologies that synchronise cloud-native and DevOps processes is growing quickly.

Here are some things to consider to implement Cloud Native DevOps simpler:



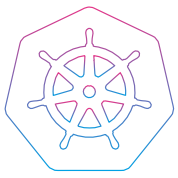
1. Containerisation

Eliminates implementation disputes between the operation and development teams, facilitating communication between developers and testers.



2. Switching to Microservices

Creates a set of procedures, terminology, and tools for the development and operation teams. Microservices provide complex process automation and make it simpler to move toward agile product development, which is necessary for continuous delivery.

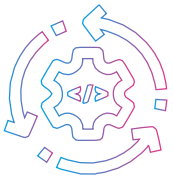


3. Using a Container Orchestration Platform like Kubernetes

Overcomes the difficulties posed by underpinning computing, storage, and networking.

4. Implementing Test Automation

Vastly simplifies the QA process while improving reliability.



5. Establishing CI/CD Pipelines

Minimises complexities and error risks. Additionally, it enables developers to concentrate on the final product rather than correcting problems.

Cloud Native DevOps Mistakes to Avoid

a. Overuse of Tools

As soon as you make the switch to cloud-native DevOps, you'll try to automate as many tasks as you can. But you can't just keep adding tools to achieve this. You must choose the greatest tools for your application and put them together in the best possible way. You will lose a lot of time and money if you use your equipment excessively.

Another error made in this situation is when developers become overly reliant on a certain tool. However, the heart of DevOps resides in the spirit of cooperation and the application of the right procedures, which help to boost productivity and improve procedures.

b. Failing to Implement Monitoring

Testing within the pipelines for implementation streamlines things, but it also makes things incomplete and constrained. By revealing every flaw that develops even after testing, continuous monitoring, on the other hand, can optimize the entire process.

c. Lapses in Security

The time and money required for security inspections might be significant. Teams frequently imagine that the CI/CD workflows make use of security checking capabilities. A crucial step for DevOps to take to protect against process vulnerabilities in the implementation of a separate tool that handles security.

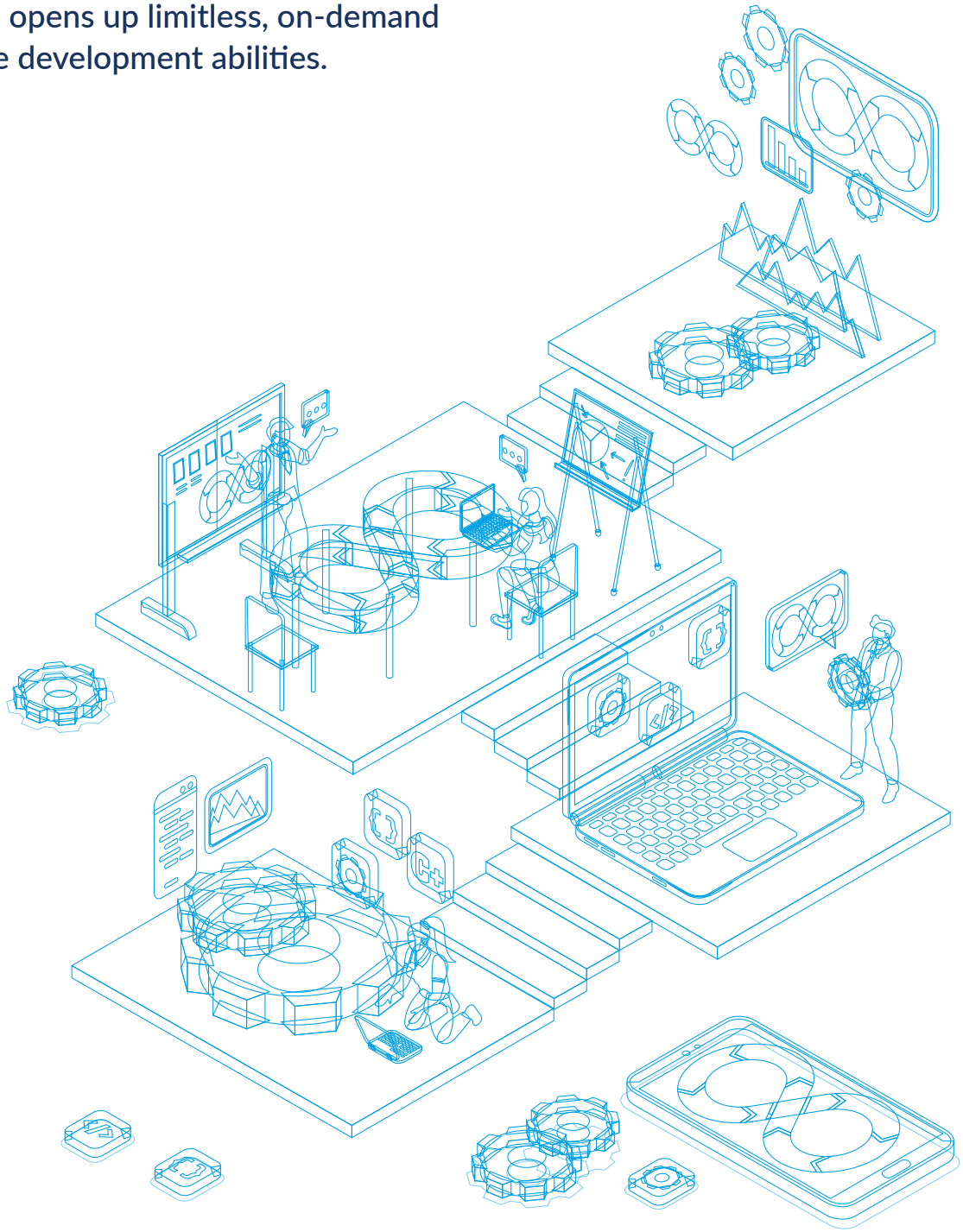
d. Being Disorganised

DevOps adoption should be a gradual process that involves lots of learning along the way. It is just not viable to expect a corporation that has been utilising on-premise applications to instantly align all of its platforms and structures into a single cloud-native architecture. Making new cloud-native applications might be simple, but converting current applications will probably take some time.

Going step-by-step would be the wisest course of action. Applications should first be moved to the cloud before moving on to microservices from monoliths. After that, you can set up platforms for orchestrating containers. But none of this will work until you have the culture change we always preach about.

Conclusion

Achieving Cloud Native DevOps Maturity aids companies in bringing innovative technologies to market more quickly, accelerating their digital transformation. The cloud-native strategy has been a boon for businesses that produce software since it reduces massive cloud expenses while increasing efficiency and performance. Utilising the full potential of cloud-native DevOps opens up limitless, on-demand software development abilities.



Get in touch with Codification to start your journey toward Cloud Native DevOps Maturity

Visit our website to learn more: www.codification.io/services

Services we offer:

- Cloud Native Application Development
- Cloud Platform Building and Migration
- Enterprise Transformation
- Security and DevSecOps
- Governance and Compliance

About Codification

Codification is a Cloud Native transformation consultancy, with a team of over 100 engineers, consultants and business professionals distributed across the world. We were founded in 2019 in the United Kingdom. We have grown since then to have a presence in Europe, the Middle East and Asia, serving leading multinational corporations, government institutions, global banks, and industry giants with our consultancy and expertise.

Through our experience, we have noticed that visionary leaders want to transform their organisations into technology companies in order to thrive in the new digital-first economy. Here, businesses want to release software faster, improve quality and build a continuous improvement culture where the best ideas win. At Codification, we establish the direction of a company's technological transformation journey and help implement new technologies and processes, resulting in a modernised digital-ready organisation.

CODIFICATION

Codification United Kingdom
The Core
Bath Lane
Newcastle upon Tyne
NE4 5TF

Phone: +44 01670 223994

Web: www.codification.io/